Домашний компьютер Правец 8Д

ОРЛИН ВЫЛЧЕВ
инж. ПЕНЧО СИРАКОВ
ДИМИТР ВАВОВ

Домашний компьютер Правец 8Д

ГОСУДАРСТВЕННОЕ ИЗДАТЕЛЬСТВО „ТЕХНИКА"
СОФИЯ, 1988

УДК 681.3.06

Книга является в помощь при работе с домашним компьютером Правец-8Д. Приведены указания по включению компьютера, раскрыты его технические и программные возможности. Основная часть посвящена описанию команд и функций языка БЕЙСИК для Правец-8Д вместе с иллюстрирующими программами. В приложениях предлагаются справочные данные.

Книга расчитана на всех читателей, которые уже сделали свои первые шаги в программировании и хотят использовать компьютер для игр, в качестве помощника в овладевании новыми знаниями, а также как средство развертывания творческих возможностей.

Перевод сделан по изданию:
Орлин Вълчев, Пенчо Сираков, Димитър Вавов
Домашен компютър Правец-8Д
Държавно издателство "Техника", София, 1986

681

ПРЕДИСЛОВИЕ

В последние годы мир был буквально залит волной микрокомпьютеров - очередным чудом микроэлектроники. Доступная цена, небольшие размеры, повышенная надежность и простота обслуживания - вот причины их массового появления в институтах, производственных предприятиях, школах и частных домах.

Постепенно сложились три основные группы микрокомпьютеров:

- персональные компьютеры универсального назначения;
- микрокомпьютеры для специализированных промышленных и научно-исследовательских целей;
- домашние компьютеры для обучения, игр, развлечений и помощи в ведении домашнего хозяйства.

Представителем последней группы является первый болгарский домашний компьютер Правец-8Д, самый младший представитель семейства микрокомпьютеров Правец, производимых Комбинатом микропроцессорной техники в г. Правец. В отличие от персональных компьютеров Правец-82 и Правец-16, которые были созданы в Институте технической кибернетики и роботики Болгарской академии наук, домашний компьютер является собственной разработкой и внедрением комбината. Цифра 8 показывает разрядность микропроцессора, а буква Д означает домашний.

Как типичный представитель современных домашних компьютеров Правец-8Д предлагает функционально развитую версию популярного языка программирования БЕЙСИК, оперативную память с объемом 65536 символов (около 40 страниц текста), три звуковых канала, графику с высокой разрешающей способностью и 8 цветов. Предусмотрена возможность подключения печатающего устройства и дискового устройства (на гибком магнитном диске), что позволяет использовать компьютер для корреспонденции, обработки документов и других профессиональных целей.

Определение "домашний" утвердилось для самых маленьких компьютеров. Для них типично использование бытового кассетофона в качестве внешней памяти и домашнего телевизора в качестве монитора, чем определяется такой важный показатель как доступность цены. Кассетофон, однако, является специализированным устройством для записи и воспроизведения музыки, а не для работы с данными, поэтому его использование вместо дискового устройства

представляет собой компромисс. По существу, это предопределяет и цели, для которых можно эффективно использовать домашний компьютер:

- обучение программированию и различные учебные дисциплины, языковая подготовка;
- игры и развлечения (попасть в компьютерное царство легче всего через чудесный мир игр);
- использование в быту, решение личных задач.

До сих пор компьютеры попадали главным образом к профессионалам - программистам, инженерам и т.д. Домашний компьютер впервые становится на самом деле частью бытовой техники, но наиболее сложной и взыскательной при работе в нашем доме.

Компьютеры, подобные Правец-8Д, привлекают главным образом детей. Достаточно только видеть как маленькие группы школьников терпеливо дожидаются свободного места в компьютерных клубах. А когда и их учителя являются не менее увлеченными людьми, успех на самом деле гарантирован. Один готовый задержаться и после уроков энтузиаст, способный оживить школьные компьютеры, может дать тот начальный толчок, в результате которого появятся десятки новых программистов.

Первый болгарский домашний компьютер призван стать экономически оправданным средством для обучения и практических занятий по программированию, средством достижения компьютерной грамотности.

1. ПОДГОТОВКА К РАБОТЕ

Весь компьютер Правец-8Д собран в одном корпусе с размерами приблизительно 35 на 25 cm. Достижения микроэлектроники позволили сконцентрировать всю вычислительную мощность в одном чипе - микропроцессоре. Очень маленький объем занимает и оперативная память. Добавлен блок питания, небольшой динамик и несколько выводов для связи с внешними устройствами. Все это смонтировано и подсоединено к одной плате. Микрокомпьютеры на одной плате являются сегодня образцом современной техники.

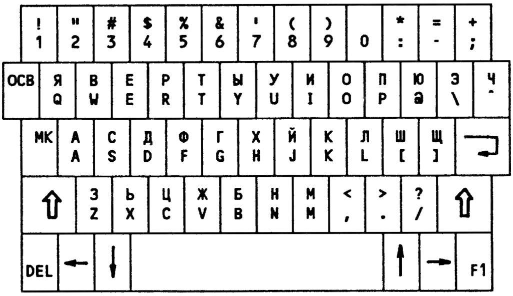
В компьютере Правец-8Д используется восьмиразрядный микропроцессор СМ630. Это означает, что данные и числа обрабатываются группами по 8 битов (1 байт). Размер адреса в один байт достаточен для адресации 65536 байтов оперативной памяти или, согласно принятому обозначению, 64К (К=1024 байта). Четвертая часть памяти выделена для версии языка БЕЙСИК. Эта память является постоянной, т.е. ее содержание не разрушается при выключении. Сам БЕЙСИК является расширенным по сравнению с известной версией для компьютера Правец-82 и включает несколько специализированных звуковых и графических команд. Подобные команды являются общепринятыми для домашних компьютеров. Современный маленький компьютер должен обладать усиленными графическими и музыкальными возможностями, чтобы более полно отвечал потребностям, связанным с обучением и играми.

Оставшаяся часть оперативной памяти обеспечивает около 48К для ввода программ. Способ распределения и использования памяти отличается от способа, который реализован в Правец-82. Это обусловлено тем, что эти два компьютера относятся к разным классам и имеют различное назначение. Программы для одного из этих компьютеров не могут непосредственно выполняться на другом.

1.1. КЛАВИАТУРА

Нераздельной частью компьютера является его клавиатура, смонтированная над основной платой. Посредством клавиатуры выдаются команды и вводятся данные. Она обеспечивает единственный способ связи между человеком и компьютером. Ниже показана клавиатура, которая с

небольшими изменениями подходит для большинства микрокомпьютеров. С помощью клавиатуры компьютера Правец-8Д можно вводить только заглавные буквы латиницы и кириллицы. Кроме букв, цифр от 0 до 9 и специальных символов, таких как . , ! + и т.д., имеется несколько специализированных клавиш:



ESC или ОСВ - управляющая клавиша "освобождение";

CTRL или МК - управляющая клавиша "машинный контроль";

DEL или ← - клавиша для стирания символов с экрана;

SHIFT или ⇧ - клавиша для выбора верхнего или нижнего регистра, в нашем случае латиницы или кириллицы;

RTN или ↵ - клавиша RETURN для перемещения маркера на следующую строку и передачи управления компьютеру.

Наиболее часто используется клавиша RETURN. Нажатие на нее обычно означает окончание работы пользователя и начало работы компьютера. Процессор любого компьютера имеет два основных режима - режим активной работы и режим ожидания. Когда компьютер закончил работу и ожидает новую команду или данные, он дает знать об этом с помощью мигающего квадратика на экране, называемого курсором (указателем, светлинным маркером). Когда пишем что-то с клавиатуры, соответствующий текст появляется на экране, а курсор смещается символ за символом. Компьютер продолжает ждать, пока не будет нажата клавиша RETURN. Если мы ввели команду, то тем самым возвращаем и управление задачей компьютеру. Курсор исчезает с экрана и появляется снова лишь после того, как компьютер закончит работу. И снова пользователь может задавать новую команду и вводить ее путем нажатия клавиши RETURN.

Имеются другие две чисто компьютерные клавиши - ESC и CTRL. Общим для них является то, что они не используются самостоятельно, а лишь в комбинации с другими клавишами.

Разница заключается в способе задания комбинации. При комбинации с ESC сначала необходимо нажать и отпустить саму клавишу ESC и лишь после этого нажать и отпустить другую клавишу. Для комбинации с CTRL процедура совсем иная - клавиша CTRL держится нажатой, пока не будет нажата вторая клавиша комбинации. Именно таким образом печатаются большие (прописные) буквы на пишущей машинке - клавиша верхнего регистра держится нажатой и нажимается клавиша соответствующей буквы. Произвольное нажатие клавиши CTRL не ведет ни к каким изменениям программы или действий компьютера.

Принято обозначать комбинацию управляющей клавиши и второй клавиши следующим образом:

CTRL-C, что означает: нажмите CTRL и, не отпуская этой клавиши, нажмите клавишу C;

ESC-C, что означает: нажмите и отпустите ESC, а затем нажмите и отпустите C.

В комбинациях с управляющими символами могут участвовать лишь латинские символы!

Смысл управляющих клавиш заключается в присвоении еще двух значений любой клавише клавиатуры. Это используется конструкторами самих компьютеров, которые встраивают стандартно определенные функции как комбинации с управляющими клавишами. Так например, комбинация CTRL-C используется почти всегда как команда прерывания выполняемой программы.

Программисты могут задавать новые комбинации и изменять существующие, естественно, без нарушения работы системы.

Осталось рассмотреть еще две специализированные клавиши, которые, однако, аналогичны по своим функциям известным клавишам пишущей машинки. Одна из этих клавиш служит для стирания символов на экране - DEL или ←. Однократное ее нажатие возвращает курсор на одну позицию назад, стирая при этом на экране и из памяти последний введенный символ. Пишущие машинки имеют подобную клавишу для возврата на один шаг назад. Второй клавишей, имеющей аналог среди клавишей пишущей машинки, является клавиша для выбора верхнего или нижнего регистра. Обозначенная на клавиатуре компьютера как SHIFT, ЛАТ или ⇧, эта клавиша служит для выбора кириллицы или латиницы, в то время как в классической пишущей машинке она используется для выбора больших или маленьких (прописных или строчных) букв в рамках одной азбуки.

В отличие от некоторых других компьютеров Правец-8Д не имеет клавиши "замыкания" верхнего регистра (известного как CAPS LOCK). Компьютер "замыкает" клавиатуру на верхний регистр посредством управляющего символа CTRL-T.

Чтобы закончить со специфичными клавишами Правец-8Д, следует отметить четыре клавиши со стрелками, расположенные по две с обеих сторон клавиши пробела. Они служат для перемещения курсора на экране в четырех направлениях, без изменения при этом содержания памяти и, соответственно, изображения на экране. Эти клавиши особенно удобны для перемещения объекта на экране и интенсивно используются при редактировании.

Обратите внимание, что клавиша стирания последнего введенного символа может быть обозначена левой стрелкой. От клавиши с левой стрелкой для движения курсора она отличается как функцией, так и размером и расположением на клавиатуре. Если вы работали на Правец-82, то наверное заметили, что на некоторых моделях этого компьютера имеются левая и правая стрелки, которые используются для редактирования текста на экране. Стрелки в Правец-8Д невозможно использовать для стирания и ввода, они предназначены лишь для позиционирования на экране. Правой стрелкой (CTRL-V) компьютера Правец-82 в нашем случае будет комбинация CTRL-A, а левой (CTRL-H) - клавиша DEL.

Возможное сравнение с клавиатурой более мощного Правец-82 покажет отсутствие у домашнего компьютера двух специальных клавиш. Одна из них - красная клавиша RESET (или RST), которая используется для аварийного прерывания текущей программы. При этом на экране появляется курсор, программа остается в памяти в неизменном виде, а компьютер ожидает ввода новой команды. Так как случайное нажатие на RST сопряжено с нежелательным эффектом, эта клавиша умышленно сделана труднодоступной и расположена на нижней стороне корпуса Правец-8Д.

Другой клавишей, которая на самом деле отсутствует, является клавиша RPT. В случае Правец-82 ее нажатие в комбинации с другой клавишей приводит к многократному вводу выбранного символа. В домашнем компьютере не предусмотрена клавиша повторения, а обеспечено автоматическое повторение произвольного символа, если соответствующая клавиша удерживается нажатой более 2 секунд.

Максимальное число символов в строке равно 80. Если в одной строке вводится более 75 символов, то при вводе символа с очередным номером 76 будет дан звуковой сигнал. Так компьютер предупреждает об окончании строки, аналогично звонку в обычной пишущей машинке. Звуковой

сигнал прозвучит и при вводе 77-го, 78-го и 79-го символов. На месте 80-го символа на экране появится \ , а курсор переместится в начало следующей строки.

Функции комбинаций управляющих клавиш с другими клавишами даны в приложении.

1.2.ПЕРИФЕРИЯ

Для нормальной работы с компьютером необходимы некоторые внешние устройства или так называемая периферия.

В первую очередь необходимо устройство, на котором мы могли бы видеть наши команды, ход выполнения задачи, различные результаты и сообщения. Таким устройством является видеомонитор, имеющий пассивную роль в вычислительном процессе. На его экране отображается все, что вводится нами с клавиатуры, а также текущие результаты выполнения компьютерных задач.

Как уже отмечалось, домашние компьютеры работают с обычными бытовыми телевизионными приемниками. Могут использоваться приемники черно-белого изображения, но хорошие цветовые возможности и характер применения домашних компьютеров предполагают цветное изображение. Разнообразие телевизионных приемников различных производителей и различных систем может затруднить подключение их к компьютеру.

Кроме разъема для подключения телевизора компьютер Правец-8Д имеет и другой разъем - для подключения монитора цветного изображения (RGB). Телевизионные приемники "Электрон" и мониторы, специально предназначеные для работы с компьютерами, обеспечивают более качественное изображение. Работа с текстом на всей площади экрана требует четкого изображения на всех его частях. В то время как в Правец-82 связь с монитором реализована посредством дополнительной платы (модуль RGB), в Правец-8Д эта связь обеспечивается стандартно на основной плате. Необходимый кабель поставляется обычно вместе с монитором.

В дальнейшем будем предполагать, что вы уже подсоединили к вашему компьютеру подходящий телевизионный приемник цветного изображения.

Конфигурация компьютер-монитор обеспечивает возможность лишь для начальной практики по программированию - для ввода отдельных команд и программ и проверки их действия. Чтобы сохранить результаты вашего труда, необходимо внешнее устройство записи. Для небольшого и недорогого компьютера вполне разумно

использовать для этой цели обыкновенный бытовой кассетофон. Программы и данные записываются на обычную кассету и читаются с нее. Кассетофон обеспечивает последовательный метод доступа к информации. Это означает, что для того, чтобы добраться до последней программы, необходимо перемотать ту часть ленты, на которой записаны предыдущие программы. Этот метод ограничивает диапазон эффективно решаемых прикладных задач, но в то же время отлично обслуживает работу компьютера в большинстве случаев.

При работе с кассетофоном необходимо регулировать силу звука. Данные, естественно, не воспроизводятся как мелодия, при их чтении слышится лишь специфичный шум. Тем не менее правильная регулировка звука является очень важной. При слишком большой или слишком малой силе звука информация будет поступать в компьютер искаженной, что приведет к ошибкам в программе.

Так как кассетофон представляет собой основную периферию компьютера Правец-8Д, несколько следующих страниц посвящены наиболее важным процедурам и правилам работы с ним.

1.3. РАБОТА С КАССЕТОФОНОМ

Предположим, что дома у вас уже имеется какой-нибудь кассетофон. Самый распространенный и дешевый метод хранения программ для домашнего компьютера заключается в их записи на обычную кассету с магнитной лентой. Метод этот ни в коей мере не является ни самым удобным, ни самым быстрым, ни самым надежным, но все еще остается значительно более дешевым по сравнению с использованием дисковых устройств. Если все же вам необходимо приобретать что-либо специально для компьютера, покупайте дешевый кассетофон. Хорошо, если он будет иметь счетчик - это серьезно облегчит поиск программ. Стереофонические кассетофоны для использования с компьютером не нужны - компьютер работает лишь с одним каналом.

На практике предпочтительно использование одного и того же кассетофона. Довольно часто при попытке загрузить программы, написанные на другом кассетофоне, будут возникать проблемы совместимости (даже небольшие различия в настройке головок могут помешать считыванию информации с вполне нормально записанной кассеты).

Тип кабеля зависит от используемого кассетофона. Сам компьютер нуждается в DIN-разъеме с тремя или семью выводами, который соответствовал бы кассетофонному входу

компьютера. Большинство устройств для записи данных используют один DIN-выход, который служит одновременно для записи и для чтения. Некоторые из самых простых монокассетофонов предлагают выводы лишь для микрофона и наушников, что немного осложняет положение. В этом случае необходимы отдельные разъемы для входа и выхода, причем при записи используется лишь связь MIC, а при чтении - связь EAR.

Предпочтительно использование коротких компьютерных лент (от 10 до 15 минут) вместо стандартных кассет на 60 и 90 минут. Длинная лента затрудняет поиск необходимой программы. Более длинные и более тонкие ленты легче подвергаются растягиванию, при котором качество записи критически ухудшается.

После установки в кассетофон новой кассеты рекомендуем перемотать ленту в быстром режиме до конца, а затем обратно, чтобы быть уверенными, что она намотана плотно. Большинство кассет имеют начальный маркер, на который невозможно производить запись. Перед началом записи необходимо позиционировать ленту непосредственно за начальным ленточным маркером. В сущности имеет смысл оставить кассету вращаться без записи приблизительно 15 секунд, так как большинство ошибок при записи возникает из-за повреждений подверженного внешним воздействиям начала ленты (или из-за растягивания первых нескольких сантиметров). Теперь установите счетчик на ноль и можете начинать работу.

Чтобы записать программу на ленту, необходимо выполнить следующие действия:

1. Проверьте, включен ли кассетофон и поставлена ли в него чистая, позиционированная кассета.

2. Убедитесь, что связь компьютер-кассетофон установлена правильно.

3. С помощью команды LIST проверьте на экране программу, которую собираетесь записывать.

4. Введите команду:

CSAVE "*прг*",S

где *прг* - имя программы. Имя может иметь длину до 16 символов, но на практике гораздо удобнее использовать короткие и легко запоминающиеся имена.

Компьютер Правец-8Д пишет и читает в двух различных режимах. Автоматически устанавливается режим быстрой передачи со скоростью 2400 бит/сек (бод). Если прибавите S, то компьютер выберет медленный режим со скоростью 300 бит/сек. Оба метода достаточно надежны, но все-таки лучше из предосторожности сделать по одной копии более ценных программ в медленном режиме.

5. Нажмите клавиши RECORD и PLAY на кассетофоне.

6. Нажмите клавишу RETURN на компьютере. Когда завершится выполнение команды CSAVE, в текущей позиции курсора появится обычное сообщение ГОТОВ. Остановите кассетофон.

Если вы аккуратно выполнили описанные действия, то теперь у вас имеется копия программы на кассете. Рекомендуется немедленно проверить качество записи.

Полные возможности всех форматов команды CSAVE, а также способ проверки качества записи, описаны во второй части руководства.

Если вы желаете загрузить с ленты программу, которая была записана командой CSAVE, необходимо выполнить следующие действия:

1. Убедитесь, что кассетофон включен, регулятор силы звука установлен в среднее положение, а регулятор тона - в самое верхнее положение.

2. Проверьте связь кассетофон-компьютер.

3. С помощью счетчика позиционируйте ленту в начало программы, которую будете загружать.

4. Введите:

CLOAD "*прг*",S

Имя файла *прг* должно быть написано точно так, как оно было задано при записи. Используйте S только в том случае, если программа записана в медленном режиме. В противном случае компьютер не распознает программу. Если есть сомнения в точности имени, можно использовать форму

CLOAD ""

и тогда компьютер загрузит первую программу, встреченную им на ленте.

5. Нажмите клавишу RETURN на компьютере.

6. Нажмите клавишу PLAY на кассетофоне. После успешной загрузки компьютер выдает сообщение ГОТОВ.

Остановите кассетофон.

Полные возможности всех форматов этой команды и способы записи и загрузки блоков памяти и массивов описаны во второй части руководства.

1.4. УХОД ЗА КАССЕТАМИ

Работа с кассетами неизбежно создает проблемы. Потеря любой программы, на которую вы затратили свой труд, представляет истинное несчастье. Здесь даются некоторые рекомендации по сохранению кассет с программами. Следует подчеркнуть, что если лента не использована один или два месяца, существует опасность разрушения части записи.

Вот некоторые простые правила, которые помогут уменьшить этот риск:

1. Не записывайте ничего в начале ленты, на отрезке, соответствующем 10-15 секундам ее вращения. Большая часть проблем с натяжением и нарушениями покрытия возникает именно на этой части ленты.

2. Если кассета не используется в данный момент, она должна находиться в футляре.

3. Никогда не оставляйте кассету на телевизоре или другом устройстве, генерирующем электромагнитные волны. Они могут стать причиной повреждения записи.

4. Не прикасайтесь к поверхности ленты. Закончив работу, перемотайте кассету до конца, чтобы только маркер подвергался внешним воздействиям.

5. Перед записью на кассету необходимо перемотать ленту до конца, а затем вернуть ее обратно. Таким образом будет обеспечено необходимое натяжение.

6. Регулярно очищайте головки записи и чтения кассетофона.

7. Записав программу, укажите на кассете и футляре ее имя и показание счетчика. Необходимо обязательно отметить, в каком режиме записана кассета (т.е. в быстром или медленном), а также номер версии программы, если вы еще не закончили ее разработку.

8. Когда запишете последнюю версию (оригинал), защитите кассету, чтобы предотвратить случайное стирание (для этого удалите пластмассовый ограничитель на торце кассеты, после чего запись станет невозможной).

9. Делайте копии наиболее важных программ.

10. Периодически перематывайте ленты, даже если вам ничего не нужно загружать с них. Это предотвратит намагничивание соседних секций ленты.

Некоторые кассетофоны могут вызвать от компьютера фальшивое сообщение об ошибке ERRORS FOUND *после выполнения* CLOAD. *Вариации ведущего сигнала ленты могут "обмануть" встроенные в компьютер средства контроля, в результате чего будут зарегистрированы ошибки. Вопреки сообщению об ошибке программа загрузится правильно, однако ее автоматический запуск после команды* CSAVE AUTO *будет невозможным.*

1.5. ДРУГАЯ ПЕРИФЕРИЯ

Кроме телевизора и кассетофона компьютер Правец-8Д может работать и с другими периферийными устройствами.

В первую очередь это печатающее устройство (принтер). Для него предусмотрен специальный разъем, а на основной плате имеется встроенный параллельный интерфейс типа CENTRONICS. Это означает возможность работы со всеми устройствами, поддерживающими этот интерфейс, в частности с распространенными матричными печатающими устройствами, производимыми в г.Петрич, НРБ, или японской фирмой EPSON. Важно, что Правец-82 и Правец-8Д соблюдают единый стандарт при кодировании символов. На практике это означает, что любое печатающее устройство для Правец-82 со стандартной кириллицей можно использовать с Правец-8Д при наличии параллельного интерфейса, причем без специальной адаптерной платы, а подключенное только посредством кабеля. Схема выводов кабеля дана в конце книги.

На задней панели компьютера имеется еще один разъем, называемый разъемом расширения. Он имеет универсальный характер и позволяет подключение дискового устройства, модема для передачи данных на расстояние, управляющих ручек для игр (джойстик). К сожалению, самое интересное устройство - дисковод, довольно редко используется в домашних компьютерах. Эти устройства предлагаются небольшим числом производителей в мире, цена их высокая, зачастую даже более высокая, чем цена самого компьютера. Дешевые домашние компьютеры потому и дешевы, что не работают с дисковыми устройствами. Это, естественно, отражается на удобствах работы с ними.

1.6. ВКЛЮЧЕНИЕ КОМПЬЮТЕРА

Компьютер подключается к электросети со стандартным напряжением 220 V. Предварительно необходимо подсоединить к компьютеру телевизионный приемник и кассетофон. Схема выводов домашнего компьютера дана в приложении.

В первую очередь необходимо включить телевизор и дать ему нагреться. Затем настройте его на 3-й или 36-й канал. Если затем включите и компьютер, на экране появится светлая надпись:

```
МИКРОКОМПЮТЪР ПРАВЕЦ-8Д
  BASIC БРВ

37631 СВ. БАЙТА

ГОТОВ
■
```

Прежде чем начать работу, отрегулируйте яркость и контрастность, чтобы получить на экране хорошее изображение. Возможно вам придется при этом переключиться на другой канал.

После указания модели компьютера, версии интерпретатора БЕЙСИК и аббревиатуры разработчика (БРВ - База развития и внедрения комбината в г.Правец), компьютер сообщает об объеме свободной памяти в байтах (т.е. в символах).

Появившееся на экране слово ГОТОВ означает, что компьютер готов к работе и ожидает инструкций. Мигающий квадратик является маркером, показывающим позицию, в которую будет введен или из которой будет выведен следующий символ.

Если вы готовы приступить к работе, желаем вам успеха!

2. КОМАНДЫ И ФУНКЦИИ

В этом разделе описано использование всех зарезервированных слов в версии языка БЕЙСИК для Правец-8Д. Многие команды и функции иллюстрируются примерами.

В описании приняты следующие обозначения:

прг - имя программы (имя файла);
усл - логическое условие
выр - выражение
оп - оператор
а - адрес
n,m - числа
x,y - координаты
s - символьная цепочка
v - переменная

Для некоторых команд и функций используются и другие специфичные символы, пояснения к которым даются в самом тексте.

Все, что обозначено в формате мелким курсивом, при использовании команд и функций заменяется конкретными значениями. Обозначения, которые даны заглавными буквами, остаются без изменения.

Те части команд и функций, которые даются в квадратных скобках, не являются обязательными.

Стандартными для языка БЕЙСИК являются символы: # - для обозначения шестнадцатеричных чисел, % - для целых чисел, $ - для символьных цепочек (строковых данных).

Когда программы вводятся в память, интерпретатор БЕЙСИК замещает каждое зарезервированное слово специальным кодом (token), занимающим один байт. Эти коды даны после формата каждой команды или функции.

Правила работы на языке БЕЙСИК дают достаточную свободу при конкретизации отдельных параметров и аргументов в командах и функциях. Если явно не указано другое, в любом из описанных полей могут задаваться не только числа и переменные, но и выражения.

Работа с компьютером постепенно формирует необходимые практические навыки использования отдельных групп команд и функций.

2.1. КОММЕНТАРИЙ

REM

Формат: REM
Код: 157

Ввод комментария к программе осуществляется командой REM. Ограничений на содержание комментария нет, так как он пренебрегается интерпретатором.

Включение REM увеличивает занимаемую программой память. Независимо от этого рекомендуется включать в программу комментарий, чтобы сделать ее более легко читаемой. В первую очередь это относится к программистам - непрофессионалам. Совершенно ясная сегодня программа завтра забудется и разобраться в ней будет трудно, поэтому комментарии на самом деле экономят ценное время программистов.

Профессиональные программисты поступают следующим образом: подготавливают два варианта программы - один легко читаемый, с комментариями и пояснениями, и другой - более компактный, без комментариев.

Вместо REM для краткости можно использовать апостроф.

```
10 REM - - - REM - - -
20 'ЭТО ДРУГОЙ ВИД REM-ОПЕРАТОРА
```

2.2. ОПЕРАЦИИ НАД ЦЕЛЫМИ ПРОГРАММАМИ

NEW

Формат: NEW
Код: 193

Эта команда стирает в памяти текущую программу и ее переменные. На практике она очищает доступную пользователю RAM-память.

Рекомендуется в начале каждой новой программы однократно использовать NEW. Это гарантирует вам, что в памяти не останется "мусора" - инструкций и данных из предыдущей программы.

CLEAR

Формат: CLEAR
Код: 189

Команда стирает значения всех используемых в данный момент переменных. В следующем примере на экран выводятся значения пяти переменных - трех числовых и двух символьных. Строка 100 стирает переменные, а строки 110 и 120 выводят на экран значения их после стирания (0 для числовых переменных и пустые цепочки - для символьных переменных).

```
10 REM - - - CLEAR - - -
20 CLS
30 A=5
40 B=10
50 C=20
60 PRINT A, B, C
70 C$="ЕЦ - 8Д"
80 B$="ПРАВ"
90 PRINT B$+C$
100 CLEAR
110 PRINT A, B, C
120 PRINT B$+C$
130 REM
140 REM СМ. NEW И RUN
```

LIST

Форматы: LIST
LIST *n*
LIST *n-m*
Код: 188

Эта команда дает возможность воспроизвести на экране находящуюся в памяти программу и видеть ее операторы. На экран может быть выведена одна строка, последовательность строк или вся программа.

Рассмотрим различные форматы команды.

При использовании формата LIST на экран будет выведена вся программа. Так как любая серьезная программа значительно превышает размер экрана, надо иметь возможность останавливать отдельные фрагменты при прокручивании всей программы на экране. В Правец-8Д для этого достаточно нажать на клавишу пробела. Движение программы на экране при этом останавливается и возобновляется после повторного нажатия на клавишу пробела.

При использовании формата LIST *n* на экран будет выведена лишь строка с номером *n*, например

```
LIST 40
```

выведет на экран строку 40 программы.

Команда LIST *n-m* выводит на экран часть программы - начиная со строки *n* и до строки *m* включительно, например

```
LIST 40-80
```

выведет строки с 40 до 80.

LLIST

Формат: LLIST
Код: 142

Команда аналогична команде LIST, но отпечатывает программу на листинг печатающего устройства вместо вывода на экран.

RUN

Формат: RUN
RUN *n*
Код: 152

Использованная самостоятельно как прямая команда RUN запускает программу, находящуюся в памяти компьютера. В отличие от Правец-82 здесь после RUN не может быть указано имя программы. Команда RUN *n*, где *n* номер строки, указывает компьютеру начать выполнение программы с заданной строки. Если такой строки нет, будет выдано сообщение об ошибке UNDEFINED STATEMENT ERROR. Команду RUN можно использовать и в теле программы:

```
10 REM - - - RUN - - -
20 A$="НАЖМИТЕ КЛАВИШУ ДЛЯ КОНЦА": GOSUB
30 CLS:PAPER 1
40 X=RND (1) * 32+1: Y=RND (1) * 20+1
50 PLOT X,Y, "ПРАВЕЦ":WAIT 50
60 V$=KEY$
70 IF V$ THEN END ELSE RUN
80 Z=LEN (A$): FOR L=1 TO Z
```

```
90 ' ***ВЕРХНЯЯ СТРОКА***
100 POKE 479999+L, ASC(MID$(A$,L,1) )
110 NEXT L
120 RETURN
130 REM
140 REM CM. CONT, END, STOP
```

CLOAD

Форматы: CLOAD ""[,S]
CLOAD "*прг*"[,S]
CLOAD "*прг*",J[,S]
CLOAD "*прг*",V[,S]
Код: 182

Команда загружает программу с кассеты. Допустимы четыре формата, которые должны включать опцию S в конце, если программа была записана в медленном режиме (slow).

Команда в формате CLOAD "" загружает в память первую обнаруженную программу. Если вам неизвестны имена программ на некоторой кассете, она может быть просмотрена путем последовательной выдачи этой команды.

Команда в формате CLOAD "*прг*" загружает программу с заданным именем. Параметр "*прг*" может быть произвольным именем с длиной до 16 символов. На практике удобнее работать с более краткими и легко запоминающимися именами.

Пока идет поиск программы, компьютер уведомляет об этом сообщением SEARCHING ... (ищу), а когда программа найдена и загружается - сообщением LOADING (загружаю). За именем программы следует B, что означает файл на языке БЕЙСИК, а за блоками памяти - C, файл на машинном языке.

Обнаруженные файлы, которые по какой-либо причине не могут быть загружены, отмечаются сообщением FOUND...FILENAME (найдена программа *прг*), тогда как для правильно загруженных файлов выдается сообщение LOADING...FILENAME (загружается программа *прг*). Если при загрузке обнаружится ошибка, то компьютер выдает сообщение ERRORS FOUND и не позволяет начать автоматическое выполнение программы, если она была записана командой CSAVE "*прг*", AUTO.

Если на кассете как отдельный файл записан блок памяти, в команде CSAVE указываются начало и конец блока. При загрузке такого файла, однако, необходимо указать только его имя.

При формате CLOAD "*прг*",J команда добавит вторую программу к концу предварительно загруженной программы. Все номера строк во второй программе должны быть больше максимального номера строки в первой программе. Если это условие не соблюдено, программа не сможет выполняться правильно из-за дублирования номеров строк. Команда в формате CLOAD "*прг*",V проверяет, правильно ли данная программа (или блок памяти) записана на кассету. Находящаяся в памяти компьютера программа записывается на кассету командой CSAVE и загружается обратно в память обычным образом, но с опцией V. (Первоначальная программа остается по-прежнему в памяти компьютера). Когда начнется загрузка, компьютер выдает сообщение VERIFYING (проверяю) в верхней строке экрана. Он будет загружать байт за байтом и одновременно сравнивать их с первоначальной программой. Если загрузка завершится успешно, в текущей позиции курсора появится сообщение 0 VERIFY ERRORS DETECTED. Это означает, что программа переписана на кассету правильно и уже может быть удалена из памяти (хотя гораздо лучше всегда сделать несколько копий одной программы). Если это сообщение не последует, загрузку можно повторить, после предварительной регулировки тона и силы звука кассетофона. Если и после нескольких попыток так и не будет получено сообщение 0 VERIFY ERRORS DETECTED, остается предположить, что запись программы была некачественной, и переписать ее на кассету снова.

Команды CSAVE и CLOAD работают в быстром режиме, если в конце к ним не будет добавлена опция S для передачи данных в медленном режиме (скорость передачи при быстром режиме 2400 бит/сек, а при медленном - 300 бит/сек). Если какая-либо программа была записана в медленном режиме, то и прочитать ее можно только в этом режиме.

CSAVE

Форматы: CSAVE "*прг*"[,S]
CSAVE "*прг*",AUTO[,S]
Код: 183

Эта команда переписывает программу или блок памяти на кассету.
Наиболее часто используется формат CSAVE "*прг*", где параметр *прг* представляет собой имя файла и имеет длину до 16 символов. Эта команда переписывает текущую программу

под указанным именем. Аналогично команде CLOAD, за командой CSAVE также может следовать операнд S, означающий запись в медленном режиме. Компьютер Правец-8Д записывает одинаково надежно и в медленном, и в быстром режиме - рекомендуется делать копии более ценных программ в обоих режимах.

Программы, записанные командой CSAVE "*прг*",AUTO, после загрузки в память компьютера стартируются автоматически.

Можно записать на кассету и содержимое произвольного блока памяти. Блок задается своими начальным и конечным адресами (шестнадцатеричными или десятичными). Начальному адресу предшествует буква A, конечному - буква E. В качестве примера приведена запись на кассету текстовых и графических страниц.

Для режима HIRES:

CSAVE "*прг*",A40960,E48000

Для режима TEXT или LORES:

CSAVE "*прг*",A48000,E49119

2.3. РАБОТА С МАССИВАМИ

DIM

Формат: DIM *v*(*n*,*m*,...)
Код: 147

Команда описывает (определяет) массив данных и резервирует в памяти область для элементов массива. Это упрощает сохранение чисел или символьных последовательностей как элементов одномерного массива (вектора) или многомерного массива. Массив *v* может быть вещественным, целочисленным или символьным. Параметры *n* и *m* определяют верхнюю границу индексов, задающих размерность массива. Нижняя граница всегда ноль. Одной командой DIM можно описать несколько массивов, путем их разделения запятыми.

Примеры:

DIM P(7) - массив вещественных чисел с 8 элементами
DIM A%(3) - целочисленный массив с 4 элементами
DIM D$(2) - символьный массив с 3 элементами

Области с размером до 11 элементов могут резервироваться и без команды DIM. Присвоение значения одному элементу, например,

```
LET N(4)=7
```

автоматически определяет область для массива N, содержащего 11 элементов - от N(0) до N(10), и присваивает значение 7 пятому элементу N(4).

Команда DIM используется для массивов с большим числом элементов. Размерность массивов всегда задается в начале программы. Однажды описанный командой DIM массив уже не может быть изменен в рамках данной программы. В противном случае компьютер выдает сообщение об ошибке REDIM'D ARRAY.

В следующей программе команда DIM определяет массив D$ из 8 элементов. Каждому элементу присваивается значение с помощью DATA и READ. В конце эти значения выводятся на экран командой PRINT.

```
10 REM - - - DIM - - -
20 DIM D$(7)
30 FOR I=1 TO 6
40 READ D$(I)
50 NEXT
60 DATA " ПРАВЕЦ-8Д"," ДЛЯ ","ИМЕР",
"  ПР","БУЧНЫЙ","АЗ"
70 FOR A=1 TO 5
80 FOR B=A TO 6
90 IF D$(A)<D$(B) THEN 110
100 T$=D$(A):D$(A)=D$(B):D$(B)=T$
110 NEXT
120 NEXT
130 FOR I=1 TO 6
140 PRINT D$(I);
150 NEXT
160 END
```

В строке 20 следующего примера определен двумерный массив целых чисел. Элементам массива присвоены значения, которые затем отпечатаны в таблице из 4 колонок и 5 строк.

```
10 REM - - - DIM'2 - - -
20 DIM N% (3,4)
30 FOR A=0 TO 3
40 FOR B=0 TO 4
50 N%(A,B)=A*10+B
```

```
60 NEXT
70 NEXT
80 FOR K=0 TO 4
90 FOR J=0 TO 3
100 PRINT N%(J,K),
110 NEXT
120 PRINT
130 NEXT
140 END
```

```
0           10          20          30
1           11          21          31
2           12          22          32
3           13          23          33
4           14          24          34
```

Длина символьной последовательности в данном массиве не должна превышать 255 символов. Теоретически максимальная размерность массива тоже равна 255, а число элементов зависит от наличной памяти. Следует помнить, что массивы быстро "съедают" память и потому их не стоит использовать без особой необходимости.

STORE

Формат: STORE *v*,"*прг*"[,S]
Код: 130

Команда записывает на кассету содержимое массива *v* как самостоятельный файл с именем *прг*. Массив должен быть описан предварительно командой DIM или принят по умолчанию как стандартный массив из 11 элементов. В противном случае компьютер выдает сообщение об ошибке OUT OF DATA. Могут записываться вещественные, целочисленные или символьные массивы.

При записи посредством команды STORE используется та же процедура, что и в случае CSAVE. Стандартная скорость быстрой записи 2400 бит/сек (к команде добавляется опция S). На экране появляется сообщение SAVING *прг*, за которым следует буква, определяется тип файла: R - для реальных чисел; I - для целых чисел; S - для строковых данных (символьные последовательности). В следующей программе показано, как используются STORE и RECALL. Задается размерность массива A$ и его элементам присваиваются

значения для демонстрации. После записи массива, переменные обнуляются. В конце программа восстанавливает массив и отпечатывает некоторые значения.

```
10 REM - - - STORE - - -
20 DIM A$(20)
30 FOR F=0 TO 20
40 A$(F) = "НОМЕР  "+STR$(F)
50 NEXT
60 PRINT "ВКЛЮЧИТЕ КАССЕТОФОН НА ЗАПИСЬ."
70 PRINT "НАЖМИТЕ КЛАВИШУ.":GET A$
80 STORE A$,"МАССИВ",S
90 CLEAR
100 DIM A$(20)
110 PRINT "ПЕРЕМОТАЙТЕ КАССЕТУ."
120 PRINT "ВКЛЮЧИТЕ КАССЕТОФОН НА ВОСПРОИЗВЕДЕНИЕ."
130 PRINT "НАЖМИТЕ КЛАВИШУ." :GET A$
140 RECALL A$,"МАССИВ",S
150 PRINT A$(9)
160 PRINT A$(10)
170 END
180 REM
190 REM CM. CLOAD, CSAVE, DIM, RECALL
```

RECALL

Формат: RECALL *v*,"*прг*"[,S]
Код: 131

Эта команда является обратной команде STORE. Она загружает с кассеты содержимое файла *прг*, предварительно записанного командой STORE. Область, в которую загружается файл, определяется массивом *v*. Процедура та же самая, что и для команды CLOAD. Прежде чем использовать RECALL, следует задать размерность массива командой DIM, в противном случае будет выдано сообщение OUT OF DATA. Файл должен быть того же типа, что и массив (вещественный, целочисленный или символьный), с теми же размерами или большим по размеру.

Низкая скорость передачи данных, записанных на кассете, задается прибавлением опции S. Ту же скорость нужно было использовать и в команде STORE. Пример приведен в описании команды STORE.

2.4. ЛОГИЧЕСКИЕ ФУНКЦИИ И КОНСТАНТЫ

AND

Формат: *усл1* AND *усл2*
Код: 209

Функция сравнивает результаты выполнения *усл1* и *усл2* путем их логического умножения (И). Принимает значение -1 (истина) только при выполнении обоих условий, в противном случае ее значение равно 0 (ложь). Например, если U=10, V=9 и W=7, результат логического действия U>V AND W<8 будет равен -1.

Можно исползовать AND и в команде IF...THEN.

OR

Формат: *усл1* OR *усл2*
Код: 210

Функция OR сравнивает результаты выполнения *усл1* и *усл2* путем логического сложения (ИЛИ). Принимает значение -1 (истина), если выполнено хотя бы одно из условий. В противном случае ее значение равно 0 (ложь). Например, если U=10, V=9 и W=7, результат логического действия U<V OR W<8 будет равен -1. Функция OR может быть частью оператора IF...THEN, например:

```
20 IF (A=0) OR (B=1) THEN GOTO 300
```

Если хотя бы одно условие является истиной, осуществляется безусловный переход к программной строке 300.

NOT

Формат: NOT *усл*
Код: 202

Логическая функция NOT осуществляет логическое отрицание. Она меняет значение логического выражения на противоположное - истину (-1) на ложь (0) и обратно.

```
10 REM - - - NOT - - -
20 CLS:A=1
```

```
30 INPUT "ЗАДАЙТЕ ЦЕЛОЕ ЧИСЛО";N
40 REPEAT
50 A=A+1
60 IF NOT(A=N) THEN PRINT A
70 WAIT 20:CLS
80 UNTIL A=N
90 PRINTA:EXPLODE
99 REM
100 REM CM. AND, IF, OR
```

TRUE

Формат: TRUE
Код: 239

Системная константа со значением -1. Представляет собой результат вычисления условного выражения, являющегося истиной. Значение лжи равно 0 (см. FALSE). В сущности, все, отличное от нуля, может рассматриваться как TRUE. Очень часто значение TRUE принимается равным 1. Однако в Правец-8Д значение константы TRUE определено равным -1, т.е. дополнительному коду числа 11111111.

Внимание: логическая проверка TRUE=1 дает 0 (FALSE), в то время как TRUE=-1 дает (TRUE). Обе константы - TRUE и FALSE - используются в комбинации, чтобы сделать программу более ясной, особенно при работе с флагами. В программировании принято называть флагами такие биты, значение которых равно 0 или 1 и показывает выполнение одного из двух возможных условий.

Следующая программа демонстрирует использование TRUE и FALSE для управления циклом REPEAT...UNTIL.

```
10 REM - - - TRUE - - -
20 FLAG=FALSE
30 PRINT "ВВЕДИТЕ ЧЕТЫРЕХБУКВЕННОЕ СЛОВО"
40 REPEAT
50 INPUT A$
60 IF LEN(A$)=4 THEN FLAG=TRUE
70 UNTIL FLAG=TRUE
80 REM
90 REM CM. FALSE
```

FALSE

Формат: FALSE
Код: 240

Логическая системная константа, которая имеет значение 0 - ложь. Используется вместе с логической константой TRUE (истина).

```
10 REM - - - FALSE - - -
20 FOR A=11 TO 20
30 IF A<13 THEN PRINT TRUE ELSE PRINT FALSE
40 NEXT
50 END
```

TRUE и FALSE делают более наглядным результат оценки данного логического выражения.

Например программа

```
10 REM - - - FALSE'2 - - -
20 A=10:B=1
30 REPEAT
40 PRINT (A>B):B=B+1
50 UNTIL FALSE
60 REM
70 REM CM. TRUE
```

отпечатает 9 раз -1 (истину), прежде чем условие A>B в строке 40 перестанет быть истинным. Когда A станет равно B (10=10), компьютер начнет выводить на печать бесконечный ряд нулей. Возможно заменить в строке 50 константу FALSE нулем, но ее использование более наглядно иллюстрирует ход выполнения программы.

2.5. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

ABS

Формат: ABS(*n*)
Код: 216

Вычисляет абсолютное значение аргумента *n*. Например, ABS(-21) равно 21. Эта функция используется, когда необходимо обеспечить положительное значение результата. Например, результат выражения (A-B) будет положительным до тех пор, пока A больше B, а ABS(A-B) всегда будет положительным числом.

```
10 REM - - - ABS - - -
20 A=COS(PI)
30 IF A=-1 THEN PRINT "COS(ПИ)=":A
40 IF ABS(A+1)<1E-9 THEN PRINT
"COS(ПИ)=";A;"СОГЛАСНО ПРАВЕЦ-8Д";
50 PRINT "   (";A;"=-1)"
60 REM
70 REM СМ. SGN

COS(ПИ)=-.999999999 СОГЛАСНО ПРАВЕЦ-8Д
(-.999999999=-1)
```

SGN

Формат: SGN(*n*)
Код: 214

Выдача знака произвольного числового выражения. Функция SGN принимает следующие значения:

-1, если $n<0$;
 0, если $n=0$;
 1, если $n>0$.

В следующей программе функция SGN используется для определения знака введенного числа. К результату добавляется 2, чтобы выйти из оператора ON...GOSUB в строке 40 на соответствующую подпрограмму, выводящую на печать сообщение о знаке числа.

```
10 REM - - - SGN - - -
20 INPUT "ВВЕДИТЕ ЧИСЛО";N
30 PRINT "ЧИСЛО БЫЛО ";
40 ON SGN(N)+2 GOSUB 100,200,300
50 PRINT "НАЖМИТЕ КЛАВИШУ ДЛЯ ПОВТОРЕНИЯ С ДРУГИМ
ЧИСЛОМ"
60 GET M$
70 GOTO 20
100 PRINT "ОТРИЦАТЕЛЬНОЕ."
110 RETURN
200 PRINT "НОЛЬ."
210 RETURN
300 PRINT "ПОЛОЖИТЕЛЬНОЕ."
310 RETURN
399 REM
400 REM СМ. ABS
```

INT

Формат: INT(*n*)
Код: 215

Определяет наибольшее целое число, меньшее или равное *n*. фактически INT превращает произвольное вещественное число в целое.

Например, программа

```
10 REM - - - INT - - -
20 Z=INT(10.747)
30 PRINTZ
```

выдает 10. Необходимо внимательно работать с отрицательными числами. Например,

```
10 REM - - - INT'2 - - -
20 A=-10.56
30 PRINTINT(A)
```

дает уже -11.

SIN

Формат: SIN(*n*)
Код: 227

Вычисляет синус угла, заданного параметром *n* в радианах. Превращение градусов в радианы и обратно выполняется с использованием функции PI. Градусы умножаются на коэффициент PI/180 для преобразования в радианы, а радианы - на коэффициент 180/PI для преобразования в градусы.

Первая строка следующей программы дает значения синуса углов от 0 до 360 с шагом 10. Строка 30 превращает градусы в радианы, а строка 40 выводит на экран угол в градусах и синус этого угла.

```
10 REM - - - SIN - - -
20 FOR A=0 TO 360 STEP 10
30 RAD=A*PI/180
40 PRINT A,SIN(RAD)
50 NEXT
60 REM
```

```
70 REM CM. ATN, COS, PI, TAN
```

Результат изменяется в диапазоне между -1 и 1, не достигая ни одного из этих значений, из-за неточности многократных вычислений. Если замените строку 40, как показано ниже, результат будет округляться до пятого десятичного знака:

```
10 REM - - - SIN'2 - - -
20 FOR A=0 TO 360 STEP 10
30 RAD=A*PI/180
40 PRINT A,INT(SIN(RAD)*1E4+0.5)/1E4
50 NEXT
```

Следующая программа использует TAB в комбинации с SIN. Она вычисляет позицию для вывода звездочек, которыми вычерчивается на экране график функции. Так как значение SIN изменяется от -1 до 1, выражение 18*SIN(T) в строке 30 принимает значения от -18 до +18, которые прибавляются к позиции TAB, чтобы изменить расположение звездочки на экране.

```
10 REM - - - SIN'3 - - -
20 FOR T=0 TO 8*PI STEP PI/8
30 PRINT TAB(20+18*SIN(T) );"*"
40 NEXT
```

```
                         *
                              *
                                    *
                                        *
                                         *
                                        *
                                    *
                              *
                        *
                  *
             *
         *
        *
         *
             *
                   *
                         *
                               *
                                    *
                                       *
                                         *
                                       *
                                     *
                              *
```

COS

Формат: COS(*n*)
Код: 226

Вычисляет косинус угла, заданного аргументом *n* в радианах. Коэффициенты преобразования радианов в градусы и градусов в радианы даны при описании функции SIN. Следующий пример демонстрирует различия графиков синуса и косинуса:

```
10 REM - - - COS - - -
20 HIRES : PRINT : PRINT : PRINT
30 INPUT "КОСИНУС ИЛИ СИНУС (COS/SIN) ";A$
40 INPUT "ЗНАЧЕНИЕ (1 ДО 99) ";V
50 CURSET 0,100,3 : DRAW 239,0,1
60 FOR A=40960 TO 49079 STEP 40
70 POKE A,INT(RND(1)*2)+16
80 NEXT
90 FOR A=-PI TO PI STEP .02
100 IF A$="COS" THEN B=COS(A)
110 IF A$="SIN" THEN B=SIN(A)
120 CURSET A*38+120, (B*V)+99,1
130 NEXT
140 PRINT A$ " ГРАФИК" : WAIT 100
150 INPUT "СТИРАНИЕ ЭКРАНА Д/Н";M$
160 IF M$="H" THEN 30
170 GOTO 20
180 REM
190 REM СМ. ATN, PI, SIN И TAN
```

TAN

Формат: TAN (*n*)
Код: 228

Вычисляет тангенс угла, заданного аргументом *n* в радианах. Результат эквивалентен SIN(*n*)/COS(*n*). Для пересчета градусов в радианы и обратно см. описание функции SIN.

Следующая программа выдает значения функции TAN для углов от 0 до 360 .

```
10 REM - - - TAN - - -
20 DEF FN R(GR)=GR*PI/180
30 FOR D=0 TO 360
```

```
40 P=D/20: P%=D/20
50 PRINT D,TAN(FN R(D))
60 IF P=P% THEN WAIT 100
70 NEXT
80 REM
90 REM CM. ATN, COS, SIN, PI
```

ATN

Формат: ATN(*n*)
Код: 229

Вычисляет значение угла в радианах, тангенс которого равен аргументу *n*. Допустимые значения изменяются от -П/2 до П/2. Так ATN(1) равен П/4 или 0,785398162.

```
10 REM - - - ATN - - -
20 HIRES
30 INPUT "КООРДИНАТЫ" (X1,Y1,X2,Y2)";X1,Y1,X2,Y2
40 CURSET X1,Y1,3
50 DRAW X2,Y2,1:WAIT 100
60 AN=ATN ((Y2-Y1)/(X2-X1)):TEXT
70 PRINT "УГОЛ "AN"РАДИАН"
80 PRINT "    ИЛИ"AN*180/PI"ГРАДУСОВ"
90 END
100 REM
110 REM CM. COS, PI, SIN И TAN
```

SQR

Формат: SQR (*n*)
Код: 222

Вычисляет корень квадратный из *n*. Аргумент *n* может быть арифметическим выражением, но должен иметь положительное значение, в противном случае будет выдано сообщение об ошибке ILLEGAL QUANTITY.

Следующая программа выводит числа от 30 до 20 и их квадратные корни.

```
10 REM - - - SQR - - -
20 FOR N=30 TO 20 STEP-1
30 PRINT N,SQR(N)
40 NEXT N
```

```
30      5.47722558
29      5.38516481
28      5.29150263
27      5.19615243
26      5.09901951
25      5
24      4.89897949
23      4.79583153
22      4.69041576
21      4.5825757
20      4.47213595
```

EXP

Формат: EXP(*n*)
Код: 225

Функция вычисляет e^n, где e=2,7183. Часто используется в комбинации с функцией LN, так как EXP(LN(*n*)) дает значение антилогарифма.

```
10 REM - - - EXP - - -
20 HIRES
30 CURSET 30,30,0
40 DRAW 0,150,1:DRAW 190,0,1
50 FOR N=0 TO 5 STEP .025
60 X=N*40+30
70 Y=180-EXP(N)
80 CURSET X,Y,1
90 NEXT
100 END
110 REM
120 REM CM. LN,LOG
```

LOG

Формат: LOG(*n*)
Код: 232

Функция вычисляет десятичный логарифм числа *n* (в Правец-82 такой функции нет). Аргументом *n* может быть положительное число или выражение с положительным значением.

```
10 REM - - - LOG - - -
```

```
20 CLS
30 INPUT "ВВЕДИТЕ ЦИФРУ (1-9)";N
40 FOR A=1 TO 10
50 C=INT(RND(1)*N+1)
60 PRINT "ЛОГАРИФМ ОТ ";N*A+C;
70 PRINT " E ";LOG(N*A+C)
80 NEXT
90 REM
99 REM СМ. EXP, LN
```

LN

Формат: LN(*n*)
Код: 224

Эта одна из полезных функций компьютера Правец-8Д. Вычисляет натуральные логарифмы, т.е. логарифмы с основанием *e*. Обратной функцией является EXP(LN(*n*)). На операции с натуральными логарифмами распространяются те же правила, что и на операции с обычными логарифмами. Например, выражение

EXP(LN(*x*)+LN(*y*))

вычислит произведение *x* и *y*.

RND

Формат: RND(*n*)
Код: 223

Без этой встроенной функции многие компьютерные игры были бы скучными. Она обеспечивает элемент случайности, генерируя случайное число между 0 и 1 (может получиться точно 0, но не точно 1). Действие RND зависит от значения аргумента *n* в скобках. Обычно используется функция RND(1). В этом случае она определяет число между 0 и 1. Само по себе это не является особенно полезным, в чем можно убедиться путем выполнения несколько раз прямой команды PRINT RND(1).

Обычно случайные числа должны быть заключены в определенном интервале, чтобы имитировать, например, бросок игральной кости, или получить случайную координату X в диапазоне от 1 до 30 и т.д.

Если умножим RND(1) на некоторое число, мы тем самым изменим диапазон получаемых случайных значений (если умножим на 6, то расширим его от 0 до 5,999999). Если прибавим затем к результату 1 и округлим до целого числа, то мы действительно получим имитацию броска игральной кости. Следующая программа показывает два эквивалентных метода округления к меньшему. Один использует INT, а другой просто присваивает значение выражения RND(1)*6+1 целочисленной переменной N%:

```
10 REM - - - RND - - -
20 A=RND(1)*6+1
30 N%=A
40 R=INT(A)
50 PRINT R
60 PRINT N%
```

RND(0) повторяет последнее генерированное случайное число.

```
10 REM - - - RND'2 - - -
20 FOR R=1 TO 50
30 PRINT RND(0)
40 NEXT
```

Иногда необходимо несколько раз генерировать одну и ту же последовательность случайных чисел, например, при последовательном тестировании программы, использующей RND. Чтобы сравнить результаты отдельных прогонов программы, каждый раз необходимо использовать одну и ту же последовательность случайных величин. Получение повторяющейся последовательности достигается заданием отрицательного аргумента функции RND.

```
10 REM - - - RND '3 - - -
20 FOR K=1 TO 2:PRINT
30 S=RND(-4)
40 FOR F=1 TO 6
50 PRINT RND(1)
60 NEXT F
70 NEXT K
```

PI

Формат: PI
Код: 238

Встроенная константа со значением π=3,14159265. В кратком примере с использованием PI вычисляется площадь круга.

```
10 REM - - - PI - - -
20 CLS
30 FOR V=1 TO 5
40 R=INT(RND(1)*30+1)
50 A=PI*R^2
60 PRINT "ПРИ РАДИУСЕ" R "ПЛОЩАДЬ КРУГА РАВНА" A
70 PRINT : PRINT
80 NEXT
```

DEF FN

Формат: DEF FN *v*(*p*)=*выр*
DEF USR *a*
Код: 184

Команда используется для определения арифметических функций. Компьютер располагает большим набором встроенных функций, но команда DEF FN позволяет "встраивать" в программу новые функции. Так избегается необходимость переписывать многократно одно и то же выражение.

Параметр *v* является именем функции и подчиняется правилам для имен переменных. Для дальнейшего использования в программе функцию нужно вызывать по этому имени. Скобки являются обязательными, в них задается имя *p* аргумента функции. Принято называть его формальным параметром, но в действительности это обычное имя переменной.За знаком равенства следует выражение, включающее *p* (формального параметра). Выражение определяет как будет вычисляться значение функции для различных значений аргумента. В примере значение аргумента A определяется в цикле.

```
10 REM - - - DEF FN - - -
20 DEF FN M(Z)=Z/6*2
30 FOR A=10 TO 100 STEP 10
40 PRINT FN M(A)
50 NEXT
```

Используется и другой формат - DEF USR *a*, служащий для определения начального адреса *a* заданной пользователем подпрограммы на машинном языке (см. USR).

```
10 REM - - - DEF USR - - -
20 DEF USR=5120
30 FOR I=0 TO 21
40 READ A: POKE 5120+I,A
50 NEXT
60 DATA 162,9,189,12,20,157,128,187
65 DATA 202,208,247,96
70 DATA 0,#70,#72,#61,#77,#65
75 DATA #63,#2D,#38,#64
80 REPEAT
90 DUMMY=USR(0)
100 WAIT 50
110 FOR I=48001 TO 48009
120 POKE I,32 : WAIT 10
130 NEXT
140 WAIT 50
150 UNTIL FALSE
```

2.6. УПРАВЛЕНИЕ

GOTO

Формат: GOTO *n*
Код: 151

Нормальная последовательность выполнения программы на языке БЕЙСИК определяется номерами операторов - от самого младшего к самому старшему. Команда GOTO прерывает эту последовательность и передает управление оператору с номером *n* (поэтому называется командой безусловного перехода).

Почти во всех серьезных книгах подчеркивается, что частое использование операторов GOTO является признаком недостаточно продуманной структуры программы. Многократное включение GOTO создает настоящий хаос в логике программы и затрудняет обнаружение ошибок.

Вместо абсолютного номера строки *n* в команде может быть указана числовая переменная или выражение, например:

```
GOTO A

GOTO A+B
```

Приведем для иллюстрации еще одну небольшую программу:

```
10 REM - - - GOTO - - -
20 CLS:GOTO 40
30 END
40 GOTO 1000
50 PRINT "ПРОГРАММА ";
60 GOTO 90
70 PRINT "ПЛОХОЕ "
80 GOTO 200
90 PRINT "ДЕМОНСТРИРУЕТ ";
100 GOTO 70
200 PRINT "ИСПОЛЬЗОВАНИЕ ";
210 GOTO 1020
300 PRINT "GOTO!"
310 GOTO 30
1000 PRINT:PRINT:PRINT"ЭТА ";
1010 GOTO 50
1020 PRINT "КОМАНДЫ ";
1030 GOTO 300
2000 REM
2001 REM CM. GOSUB, ON
```

GOSUB

Формат: GOSUB *n*
Код: 155

Одна из наиболее полезных команд в языке БЕЙСИК. Подобно GOTO передает управление строке *n* - вызывает безусловный переход. В отличие от GOTO команда GOSUB передает управление временно, пока не будет выполнена данная подпрограмма. Подпрограмма, которой передается управление, должна заканчиваться оператором RETURN, возвращающим управление в основную программу. Адрес возврата записывается при выполнении команды GOSUB в специальную память компьютера, называемую стеком (см. описание POP).

Нормальную последовательность передачи управления между основной программой и несколькими подпрограммами можно нарушать лишь оператором POP, который работает непосредственно с содержимым стека и убирает из него верхний адрес возврата.

Вместо номера строки в команде GOSUB может быть задана переменная или арифметическое выражение.

Правильное использование подпрограмм может значительно увеличить эффективность и наглядность программы.

```
10 REM - - - GOSUB - - -
20 CLS:C=0
30 PRINT "ВВЕДИТЕ ЦЕЛОЕ ЧИСЛО МЕЖДУ 1 И 22"
40 INPUT N
50 IF N>22 OR N<1 THEN GOSUB 250
60 IF N<>INT (N) THEN GOSUB 250
70 IF C=1 THEN 20
80 GOSUB 100
90 PRINT F:GOTO 310
100 REM ** ПОДПРОГРАММА **
110 IF N<>1 THEN 140
120 F=1
130 GOTO 180
140 N=N-1
150 GOSUB 100
160 F=F*(N+1)
170 N=N+1
180 RETURN
190 REM **   КОНЕЦ **
200 GOTO 310
250 REM ** ПОДПРОГРАММА NO.2 **
260 CLS:C=1
270 PRINT "СОБЛЮДАЙТЕ ИНСТРУКЦИИ!"
280 WAIT 300
290 RETURN
300 REM **  КОНЕЦ NO.2 **
310 END
400 REM
410 REM CM. GOTO, ON, RETURN
```

ON

Формат: ON *усл* GOTO *n1,n2,...*
ON *усл* GOSUB *n1,n2,...*
Код: 180

Эту команду нужно комбинировать с GOTO или GOSUB. Она облегчает ветвление программы. Используется обычно как управляющая структура в программах с меню, где пользователю предлагается несколько возможностей, которые обрабатываются различными частями программы. Это показано в следующей программе:

```
10 REM - - - ON - - -
20 CLS
30 INPUT "ВВЕДИТЕ 1,2 ИЛИ 3";N
40 ON N GOTO 100,200,300
50 REM
100 PRINT "СТРОКА 100 ИЗ-ЗА N=" ;N:GOTO 30
150 REM
200 PRINT "СТРОКА 200 ИЗ-ЗА N=2":GOTO 30
250 REM
300 PRINT "СТРОКА 300 ИЗ-ЗА N=3":GOTO 30
399 REM
400 REM СМ. GOSUB, GOTO, RETURN
```

Если в строку 30 ввести 3, то в качестве адреса перехода будет взят номер строки, стоящий в конструкции ON...GOTO третьим, т.е. управление будет передано строке 300. Если *n*=2, то переход будет осуществлен на строку 200 и т.д.

Если *n* превысит число заданных в команде адресов перехода, то выполнение программы будет просто продолжено с оператора, который следует после строки, содержащей ON. Отрицательное значение *n* в INPUT вызовет сообщение об ошибке. Нецелые значения округляются автоматически.

IF...THEN...[ELSE]

Формат: IF *усл* THEN *оп11:оп12:...*
[ELSE *оп21:оп22:...*]
Код: 153, 201, [200]

Данная конструкция используется для управления последовательностью выполнения операции в зависимости от выполнения заданного условия. Элемент ELSE заключен в квадратные скобки, так как не является обязательным в данном формате. Операторами могут быть любые инструкции языка БЕЙСИК при условии, что они не превышают максимальной длины программной строки. Если *усл* принимает значение истина, то выполняются операторы, следующие после THEN. После этого управление передается следующей строке, а операторы, стоящие после ELSE, пропускаются. Эти операторы выполняются в том случае, если *усл* принимает значение ложь. В этом случае пропускаются операторы, следующие за THEN.

Конструкция IF...THEN...[ELSE] является одним из наиболее мощных средств языка БЕЙСИК. Использование ее поясняется следующим примером:

```
10 REM - - - IF/THEN/ELSE - - -
20 CLS
30 INPUT "ВВЕДИ СВОЕ ИМЯ";N$
40 CLS
50 INPUT "МУЖЧИНА ИЛИ ЖЕНЩИНА (М/Ж)";S$
60 CLS
70 PRINT "ПРИВЕТ! ТЫ ";
80 IF S$="M" THEN PRINT "ИНТЕЛЛИГЕНТЕН"
; ELSE PRINT"ПРЕКРАСНАЯ" ;
90 PRINT", ";N$;"!"
100 END
110 REM
120 REM CM. AND, GOTO, NOT, OR, ON
```

В конструкции IF...THEN...[ELSE] команда GOSUB должна использоваться в своем стандартном формате. Оператор GOTO можно опускать, при этом задается лишь номер строки. GOTO можно заменить THEN.

FOR...TO...[STEP]...NEXT

Формат: FOR *v*=*x* TO *y* [STEP *z*]...NEXT *v*
Код: 141, 195, [203], 144

Значительная часть работы компьютера связана с повторением простых задач. Наиболее эффективным средством для этого является организация циклов. Чаще всего для реализации подобных конструкций используется цикл FOR...NEXT.

Цикл FOR...NEXT указывает компьютеру выполнять содержащиеся в нем операции определенное число раз, например:

```
10 REM - - - FOR - - -
20 FOR A=1 TO 3
30 PRINT A
40 NEXT A
```

Этот цикл выводит на экран числа 1, 2 и 3. При первом проходе через цикл компьютер присваивает переменной A значение 1 и выводит его на экран в строке 30. Оператор NEXT A в строке 40 возвращает управление строке 20. Там

значение A увеличивается на 1 и цикл повторяется. Значение переменной увеличивается на 1 при каждом проходе цикла, пока не станет равным заданному после TO значению (проверка осуществляется при выполнении NEXT). Тогда цикл заканчивается, а выполнение программы продолжается со строки, которая следует за оператором NEXT.

Для шага, отличного от 1, необходимо использовать оператор STEP:

```
10 REM - - - FOR'2 - - -
20 FOR X=5 TO 20 STEP 5
30 PRINT X
40 NEXT
```

Этот цикл выведет на экран числа 5, 10, 15 и 20. NEXT выполнит свою задачу и при опущенном имени переменной. Указание имени все же может быть рекомендовано - этим достигается большая ясность, особенно в случае вложенных циклов.

Вполне возможно, чтобы STEP было отрицательным или десятичным числом:

```
10 REM - - - FOR'3 - - -
20 FOR A=20 TO 5 STEP -5
30 PRINT A
40 NEXT A
```

или

```
10 REM - - - FOR'4 - - -
20 FOR F=1 TO 2 STEP .2
30 PRINT F
40 NEXT F
```

Начальные и конечные значения цикла, так же, как и шаг, могут быть целыми числами, переменными или числовыми выражениями, например:

```
10 REM - - - FOR'5
20 A=56:C=PI/2
30 FOR X=2^4 TO A/3 STEP C
40 PRINT X
50 NEXT X
60 REM
70 REM CM. REPEAT, UNTIL
```

REPEAT

Формат: REPEAT
Код: 139

В комбинации с UNTIL эта команда создает цикл, указывающий компьютеру повторять серию инструкций, пока не будет выполнено определенное условие. В отличие от цикла FOR...NEXT здесь нет счетчика, нарастающего при каждом проходе. Счетчик, если он необходим, нужно программировать в цикле (как в следующем ниже примере).

Единственный способ выйти из цикла REPEAT...UNTIL до его нормального завершения предоставляется командой PULL.

```
10 REM - - - REPEAT - - -
20 HIRES:INK 4:PAPER 0
30 A=0
40 REPEAT
50 CURSET 55+A,110-A,3
60 DRAW 2+A,0,1:DRAW 0,A+A,1
70 A=A+1
80 UNTIL A>60
89 REM
90 REM CM. FOR, NEXT, PULL, UNTIL
```

UNTIL

Формат: UNTIL *усл*
Код: 140

Команда является частью задающей цикл конструкции REPEAT...UNTIL. После каждого прохода цикла вычисляется условное выражение, и как только значение его станет истиной, управление программой перейдет к следующему оператору, т.е. выйдет из цикла. Если же условие останется ложью, управление возвратится обратно к оператору, следующему за командой REPEAT, инициировавшей цикл. Если команда REPEAT не обнаружена, компьютер выдает сообщение об ошибке ?BAD UNTIL ERROR.

PULL

Формат: PULL
Код: 136

Команда PULL выбрасывает из стека верхний адрес возврата из циклов REPEAT...UNTIL. Действие аналогично действию POP для конструкции GOSUB...RETURN.

Практически PULL представляет аварийный выход из цикла и используется при попадании в неясные ситуации.

```
10 REM - - - PULL - - -
20 A=9
30 REPEAT
40 : B=A
50 : REPEAT
60 :    PRINT B;
70 :    B=B-1
80 :    IF B<0 THEN PULL:GOTO 100
90 : UNTIL B=0
100 :A=A-1
110 :PRINT
120 UNTIL A=-5
130 REM
140 REM CM. POP, REPEAT, UNTIL
```

```
9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
0
-1
-2
-3
-4
```

RETURN

Формат: RETURN
Код: 156

Эту команду нужно использовать как последний оператор в любой подпрограмме. Она указывает, что компьютер дошел до конца подпрограммы и управление должно быть возвращено оператору, следующему за командой GOSUB.

Адрес возврата RETURN может быть изменен лишь командой POP.

```
10 REM - - - RETURN - - -
20 CLS
30 PRINT "ВВЕДИТЕ РАДИУС";
40 INPUT R
50 C=2*PI*R:Z=C
60 GOSUB 200
70 CLS
80 PRINT "ДЛИНА ОКРУЖНОСТИ" Z
90 A=PI*R^2:Z=A
100 GOSUB 200
110 PRINT "ПЛОЩАДЬ"Z
120 GOTO 300
200 ' ПОДПРОГРАММА ОКРУГЛЕНИЯ
210 ' ДО 2 ЗНАКОВ
220 Z=INT(100*(Z+.005))
230 Z=Z/100
240 RETURN
300 END
399 REM
400 REM CM. GOSUB, POP
```

POP

Формат: POP
Код: 134

Команда удаляет из стека верхний адрес возврата (RETURN-адрес). При вызове подпрограммы компьютер сохраняет адрес возврата в специальной области памяти, называемой стеком. Вызов каждой следующей подпрограммы добавляет к стеку очередной адрес. Когда программа встречает RETURN, она берет из стека последний поступивший адрес.

Команда POP является единственным средством изменения строгой последовательности выполнения вложенных подпрограмм. Например, если наша программа вызывает подпрограмму A, которая в свою очередь вызывает подпрограмму B, естественный ход передачи управления предполагает, что после завершения B управление вернется к A, а после завершения A вернется к основной программе. Команда POP предоставляет возможность перехода из B

непосредственно в основную программу (компьютер игнорирует самый верхний адрес в стеке и берет следующий адрес - адрес возврата из подпрограммы A в основную программу).

Во многих отношениях POP нужно рассматривать как крайне нежелательную команду. В хорошо спланированной программе с продуманной структурой не должны возникать ситуации, требующие аварийного выхода из вложенной подпрограммы.

```
10 REM - - - POP - - -
20 FOR I=-5 TO 5:PRINT I,
30 GOSUB 100
40 PRINT:NEXT
50 END
100 GOSUB 200
110 PRINT X
120 RETURN
200 IF I>0 THEN X=LOG(I) ELSE POP
210 RETURN

-5
-4
-3
-2
-1
 0
 1          7.00890077E-11

 2          .301029996

 3          .477121255

 4          .6020599991

 5          .698970004
```

STOP

Формат: STOP
Код: 179

Команда прекращает выполнение программы, выдавая при этом сообщение с указанием строки (BREAK IN...). Программу можно запустить повторно командой CONT, за

которой следует RETURN. Команда STOP действует как END, но после END невозможно запустить программу повторно, как показано в следующем примере:

```
10 REM - - - STOP - - -
20 FOR X=1 TO 3
30 PRINT "X=";X
40 PRINT "ПРОГРАММА ОСТАНОВЛЕНА."
50 PRINT "НАПИШИ CONT ДЛЯ ПРОДОЛЖЕНИЯ."
60 STOP
70 NEXT X
80 END
90 PRINT "ЭТА СТРОКА НЕ БУДЕТ ВЫВЕДЕНА, ТАК КАК
CONT НЕ ДЕЙСТВУЕТ ПОСЛЕ END."
100 REM
110 REM СМ. END, WAIT, CONT
```

CONT

Формат: CONT
Код: 187

Эта прямая команда осуществляет повторный запуск программы после прерывания. При создании программы необходимо использование CTRL-G, чтобы остановить выполнение программы для анализа изображения на экране или состояния некоторой переменной. Во многих случаях выполнение программы должно быть продолжено с точки прерывания. В таких случаях полезна команда CONT.

Команда CONT не разрешает рестартирование, если внесены изменения в какую-либо часть программы. Используется лишь в режиме немедленной обработки. Включение ее в программу рассматривается как ошибка и вызывает прерывание.

END

Формат: END
Код: 128

Команда используется для останова программы. В отличие от STOP она не выдает сообщения о номере последней выполненной строки (BREAK IN...).

Обычно команда END является последним оператором в программе. Иногда она используется и как средство прерывания при определенном условии, например:

```
10 REM - - - END - - -
20 REPEAT
30 INPUT "ВВЕДИ ЧИСЛО";NO
40 IF NO<=0 THEN END
50 PRINT "НАТУРАЛЬНЫЙ ЛОГАРИФМ ОТ"NO"ЕСТЬ"LN(NO)
60 UNTIL FALSE
70 REM
80 REM СМ. STOP
```

2.7. ОПЕРАЦИИ С СИМВОЛЬНЫМИ ЦЕПОЧКАМИ

LEN

Формат: LEN (*s*)
Код: 233

Встроенная функция, вычисляющая количество символов в цепочке (длину символьной строки) *s*. Например,

```
10 REM --- LEN ---
20 J=LEN("ПРАВЕЦ-8Д")
30 PRINT J
```

выводит на екран 9, потому что строка "ПРАВЕЦ-8Д" содержит 9 символов, а

```
10 REM --- LEN'2 ---
20 A$="НПК ПО МПТ"
30 PRINT LEN(A$)
```

выводит 10, так как эта строка содержит 8 символов и 2 интервала.

Эта функция полезна, например, когда требуется форматировать экран, который будет содержать вводимые пользователем символьные последовательности. Длина последовательностей является величиной переменной и, естественно, заранее неизвестной. Функцию LEN можно использовать и в конструкциях FOR...NEXT, когда необходимо выполнить некоторую операцию с каждым символом последовательности. Например,

```
10 FOR I=1 TO LEN(X$)
```

Пустая или нулевая последовательность (строка) имеет нулевую длину.

LEFT$

Формат: LEFT (*s*,*n*)
Код: 244

Встроенная функция для работы со строковыми данными. Аналогично функциям MID$ и RIGHT$ эта функция дает возможность извлекать символы из данной цепочки - извлекаются левые *n* символов из цепочки *s*. Следующая программа выделит и отобразит на экране все возможные левые подцепочки из символьной цепочки "ПРАВЕЦ-8Д". Если *n* превышает количество символов в цепочке, то функция возвращает всю цепочку.

```
10 REM --- LEFT$ ---
20 A$="ПРАВЕЦ-8Д"
30 FOR I=1 TO LEN(A$)
40 B$=LEFT$(A$,I)
50 PRINT B$
60 NEXT I
70 END
80 REM
90 REM СМ. RIGHT$, MID$, LEN
```

```
П
ПР
ПРА
ПРАВ
ПРАВЕ
ПРАВЕЦ
ПРАВЕЦ-
ПРАВЕЦ-8
ПРАВЕЦ-8Д
```

MID$

Формат: MID$(*s*,*m*,*n*)
Код: 246

Функция отделяет часть символьной строки *s*, начинающуюся с символа с очередным номером *m* и содержащую *n* символов. Посредством MID$ можно извлекать группы символов, произвольно расположенные в некоторой цепочке.

Если значение *n* не указано, функция отделяет все символы после символа с номером *m*. Такой же результат получается, если указано *n*, превышающее количество символов в цепочке.

Если *m*-тый символ в строке *s* не существует, результатом выполнения MID$ будет пустая строка.

В примере MID$ используется, чтобы убрать начальные и конечные интервалы из символьной строки.

```
10 REM --- MID$ ---
20 A$="        ПРАВЕЦ-8Д          "
30 IF ASC(A$)=32 THEN A$=MID$(A$,2):GOTO 30
40 IF ASC(RIGHT$(A$,1))=32 THEN A$=MI
D$(A$,1,LEN(A$)-1):GOTO40
50 PRINT "*";A$;"*"
59 REM
60 REM CM. LEFT$, RIGHT$

*ПРАВЕЦ-8Д*
```

RIGHT$

Формат: RIGHT$ (*s*, *n*)
Код: 245

Аналогично функциям MID$ и LEFT$ эта функция выделяет из символьной строки определенные символы - *n* правых символов из строки *s*.

Например, программа

```
10 REM --- RIGHT$ ---
20 A$="ПРАВЕЦ-8Д"
30 FOR I=1 TO LEN(A$)
40 B$=RIGHT$(A$,I)
50 PRINTSPC(LEN(A$)-LEN(B$));B$
60 NEXT I
70 END
80 REM
90 REM CM. LEFT$, MID$

        Д
       8Д
```

```
          -8Д
         Ц-8Д
        ЕЦ-8Д
       ВЕЦ-8Д
      АВЕЦ-8Д
     РАВЕЦ-8Д
    ПРАВЕЦ-8Д
```

выделяет указанное число символов справа налево из строки "ПРАВЕЦ-8Д". Если *n* больше числа символов в строке, на экран выводится вся строка.

STR$

Формат: STR$(*n*)
Код: 234

Эта функция преобразует числовые значения в цепочку символов (в строку). Она является обратной функции VAL, которая превращает цепочку числовых символов в число. Можно использовать экспоненты и шестнадцатеричные числа, которые будут преобразованы в стандартные символы. Символьная строка получает первый символ, который является знаком, если число отрицательное, или остается пустой, если число положительное.

```
10 REM --- STR$ ---
20 N=12.34
30 CLS:PRINT STR$(N):WAIT 15
40 PLOT 10,1,STR$(N):WAIT 45
50 PLOT  0,1,STR$(N):WAIT 15
60 PLOT 10,8,STR$(N):WAIT 15
70 X=23.5
80 PRINT
90 PRINT STR$(X):WAIT 15
100 PLOT 9,2,STR$(X)
110 PRINT "#A3 ВЫВОДИТСЯ КАК" STR$(#A3)
120 PRINT "1.345E-4 ВЫВОДИТСЯ КАК" STR$(1.345E-4)
130 PRINT "1.23E2 ВЫВОДИТСЯ КАК" STR$(1.23E2)
```

Эта программа использует PLOT, чтобы позиционировать на экране выводимые символы (с тем же успехом можно использовать PRINT@). Функция STR$ полезна не только возможностью вывода символов посредством PLOT. Превращая данное число в символьную строку, она позволяет затем его "редактирование" посредством набора специальных функций

для обработки символьных строк (например, MID$, LEFT$ и т.д.). Она полезна и когда позиционируются символы для экранов в режиме HIRES, так как посредством CHAR можно позиционировать лишь единичные символы. Следующая программа иллюстрирует это, используя STR$ для преобразования числа в символьную строку. Затем берется последовательно каждый символ цепочки, находится его код при помощи ASC и символ позиционируется на экране. Эта же техника используется и для удаления первого символа, если число является положительным.

```
10 REM --- STR$'2 ---
20 HIRES
30 FOR A=1 TO 10 STEP .5
40 A$=STR$(A)
50 GOSUB 100
60 NEXT
70 PRINT"НАЖМИТЕ КЛАВИШУ ДЛЯ КОНЦА.":GET A$
80 TEXT:END
100 'ВЫВОД СТРОКИ В РЕЖИМЕ HIRES
110 FOR B=1 TO LEN(A$)
120 CURSET B*6+A*6,A*16,0
130 CHAR  ASC(MID$(A$,B)),0,1
140 NEXT
150 RETURN
199 REM
200 REM CM. VAL
```

VAL

Формат: VAL(*s*)
Код: 235

Функция преобразует символьную цепочку *s* в число. Первым символом преобразуемой цепочки должен быть интервал, знак минус, тире или цифра, в противном случае функция возвращает ноль. Преобразование ведется до первого нецифрового символа. Числа можно представить и в экспоненциальной форме, но обрабатываются в виде, в котором выводятся на экран. Это можно проверить, если ввести, например, 1,23E2 в следующей программе. Каждый символ, отличный от цифры, будет игнорироваться компьютером, после того как в цепочке обнаружены цифровые символы.

```
10 REM --- VAL ---
```

```
20 CLS
30 REPEAT
40 INPUT "ВВЕДИТЕ СТРОКУ": A$
50 PRINT "ЦЕПОЧКА НАЧИНАЕТСЯ С ЦИФРЫ":
60 PRINT VAL(A$)
70 PRINT
80 UNTIL A$="КОНЕЦ"
90 END
95 REM
99 REM CM. ASC, STR$
```

CHR$

Формат: CHR$(*n*)
Код: 237

Функция выдает символ, ASCII код которого задан в качестве аргумента. Аргумент *n* должен быть целым числом в интервале 0-255. Использование этой функции, однако, далеко превосходит простое воспроизведение стандартного набора символов. Даже беглый взгляд на полный список ASCII кодов раскроет некоторые потенциальные возможности CHR$. Например, следующая программа использует CHR$ для задания атрибутов, которые изменяют основной цвет и цвет фона для оставшейся части строки, на которую выводятся сообщения.

```
10 REM --- CHR$ ---
20 PRINT CHR$(131);
30 PRINT CHR$(144);
40 PRINT CHR$(140);
50 PRINT "ЖЕЛТЫЕ МИГАЮЩИЕ БУКВЫ"
60 END
70 REM
80 REM CM. ASC, HEX$, STR$ И VAL
```

Функция CHR$ используется также при печати для задания управляющих символов, инициирующих различные функции и возможности печатающего устройства.

ASC

Формат: ASC (*s*)
Код: 236

Выдает десятичный числовой ASCII код для первого символа цепочки *s*. Любой символ, доступный компьютеру ПРАВЕЦ-8Д, имеет соответствующий ASCII код. В следующем примере ASC используется для контроля ввода данных. Строка 40 проверяет, попадает ли первый символ введенной цепочки в интервал кодов между 65 и 90. Так как коды 65-90 соответствуют латинице, компьютер будет принимать только те символьные цепочки, которые начинаются с латинской буквы.

```
10 REM --- ASC ---
20 CLS
30 INPUT "ВВЕДИТЕ ПРОИЗВОЛЬНЫЕ СИМВОЛЫ";
K$:CLS:PRINT:PRINT
40 IF ASC(K$)>64 AND ASC(K$)<91 THEN PRINT K$
50 WAIT 100
60 GOTO 10
70 REM
80 REM CM. CHR$, HEX$, STR$ И VAL
```

HEX$

Формат: HEX$(*n*)
Код: 220

Функция превращает десятичное целое число *n* в его шестнадцатеричный эквивалент. Например, HEX$ (15) дает #F(символ # означает шестнадцатеричное значение). Десятичное число в скобках может быть произвольным целым числом от 0 (#0) до 65535 (#FFFF). Компьютер выдает сообщение об ошибке BAD SUBSCRIPT при попытке преобразовать в шестнадцатеричное представление отрицательное число или дробь.

2.8. ВСПОМОГАТЕЛЬНЫЕ КОМАНДЫ И ФУНКЦИИ

PEEK

Формат: PEEK (*a*)
Код: 230

Функция PEEK выводит число, записанное в памяти в одном байте по адресу *a*. Значение его будет в интервале от 0 до 255. Функция PEEK является обратной функции POKE, с помощью которой можно изменить содержание произвольного байта памяти.

```
10 REM --- PEEK ---
15 CLS
20 REPEAT
30 PRINT CHR$(20)
40 GOSUB 100
50 WAIT 50
60 UNTIL FALSE
70 END
100 IF PEEK(48039)=83 THEN PRINT@16,3
; "ЛАТИНИЦА" ELSE PRINT@16,3;"КИРИЛЛИЦА"
110 RETURN
199 REM
200 REM CM. CALL, DEEK, DOKE, POKE, USR
```

DEEK

Формат: DEEK (*a*)
Код: 231

Эта функция делает доступным значение, записанное в двух байтах памяти с адресами *a* и *a*+1. Возвращает числа в интервале 0-65535, запомненные в указанных двух байтах. Стандартный формат, который использует процессор 6502 для хранения адресов, представляет собой два байта, младшим из которых является первый. DEEK берет значение, сохраняемое во втором (старшем) байте с адресом *a*+1 и умножает на 256. К полученной величине прибавляет значение байта с адресом *a*.

При описании CALL приведен пример, в котором с помощью DEEK рассматривается содержание ROM памяти компьютера. Другим примером использования DEEK является возможность следить за текущим содержанием HIMEM посредством DEEK (#A6).

POKE

Формат: POKE *a*,*n*
Код: 185

Команда POKE позволяет программисту изменять содержимое байта памяти - она записывает значение *n* в байт с адресом *a*. Адрес должен находиться в интервале 0-65535, а значение *n* в интервале 0-255. Как адрес, так и значение *n* могут быть десятичными или шестнадцатеричными. Возможности этой команды иллюстрирует следующая программа, котороя изменяет состояние экрана.

```
10 REM --- POKE ---
20 FOR A=1 TO 5
25 N=ASC(MID$("ОРЛИН",A,1))
30 POKE 48034+A,N
40 WAIT 100
50 NEXT
60 WAIT 1000
70 CALL DEEK(#FFFA)
80 REM
90 REM CM. DEEK, DOKE, PEEK
```

DOKE

Формат: DOKE *a*,*n*
Код: 138

Эта команда записывает целое число из интервала 0-65535 в два байта памяти. Параметр *a* является адресом первого из двух байтов, в которые записывается целое число *n*. Формат записи обычный для микропроцессора 6502 - младший байт, за которым следует старший байт. Так например, для записи в память числа 770, можно использовать команду:

```
DOKE 30000,770
```

Эта команда запишет 2 в байт с адресом 30000 и 3 в байт с адресом 30001, так как число 770 представляется следующим образом: 3*256+2.

CALL

Формат: CALL *a*
Код: 191

Команда CALL передает управление подпрограмме на машинном языке, которая начинается с адреса *a*. Возврат в БЕЙСИК производится машинной инструкцией RTS (возврат из подпрограммы). Использование этой команды может привести к потере основной программы, если адрес указан неправильно и не передает управление к началу подпрограммы.

Вот пример потенциальной опасности использования этой команды и ее силы:

```
CALL DEEK(#FFFC)
```

Эта команда передает управление вектору "холодного старта" и рестартирует компьютер, как если бы он был выключен и снова включен. Подобным образом вызов адреса, возвращенного DEEK(#FFFC), приведет к "горячему" рестарту (как если бы была нажата клавиша RESET).

USR

Формат: USR(*a*)
DEF USR *a*
Код: 217

Команда USR(*a*) выполняет пользовательскую подпрограмму на машинном языке, передавая ей параметр *a*. Обеспечивает доступ к подпрограммам на машинном языке во время выполнения программы на языке БЕЙСИК.

Подпрограмма вызывается командой USR(*a*). После ее завершения управление возвращается основной программе (если вы, разумеется, завершили подпрограмму посредством RST), а результат либо выводится (PRINT USR(0)), либо присваивается переменной (A=USR(0)). Если никакие значения не должны передаваться подпрограмме в машинных кодах, ее можно вызвать только командой CALL.

DEF USR *a* определяет начальный адрес *a* пользовательской подпрограммы на машинном языке. Этот адрес используется позднее в USR(*a*).

```
10 REM --- DEF USR ---
20 DEF USR=5120
30 FOR I=0 TO 21
40 READ A: POKE 5120+I,A
50 NEXT
60 DATA 162,9,189,12,20,157,128,187
65 DATA 202,208,247,96
70 DATA 0,#70,#72,#61,#77,#65
```

```
75 DATA #63,#2D,#38,#64
80 REPEAT
90 DUMMY=USR(0)
100 WAIT 50
110 FOR I=48001 TO 48009
120 POKE I,32 : WAIT 10
130 NEXT
140 WAIT 50
150 UNTIL FALSE
```

WAIT

Формат: WAIT *n*
Код: 181

Данная команда останавливает выполнение программы на *n* сотых долей секунды. Большое значение имеет совместное использование WAIT со звуковыми командами. Таким образом можно управлять продолжительностью звука для команд PLAY, SOUND и MUSIC.

WAIT можно использовать и для введения пауз в программе, и для замедления изображения на экране. Важно отметить, что прервать действие WAIT с клавиатуры нельзя. Ожидание ввода с клавиатуры достигается использованием команд GET и INPUT.

HIMEM

Формат: HIMEM *a*
Код: 158

Команда HIMEM определяет *a* как верхнюю границу памяти, доступную программам на языке БЕЙСИК. Обычно компьютер сам распределяет наиболее эффективным образом доступную память между пользовательской программой, ее переменными и графическими страницами. При работе с подпрограммами на машинном языке необходимо, чтобы программист сам выделил для них часть памяти за счет памяти для программ на языке БЕЙСИК.

Естественно, команда HIMEM должна располагаться в самом начале данной программы, при этом всегда к ее использованию необходимо относиться очень внимательно.

```
10 REM --- HIMEM ---
20 PRINT "В ДАННЫЙ МОМЕНТ HIMEM = "DEEK (#A6)
```

```
30 GRAB
40 PRINT "ПОСЛЕ GRAB HIMEM =      "DEEK (#A6)
50 RELEASE
60 PRINT "ПОСЛЕ RELEASE HIMEM =   "DEEK (#A6)
70 HIMEM 3000
80 PRINT "ИЛИ ЛЮБОЕ ЗНАЧЕНИЕ      "DEEK (#A6)
90 RELEASE:REM СТАНДАРТНЫЙ HIMEM
100 REM
101 REM СМ. GRAB, RELEASE
```

FRE

Формат: FRE (0)
FRE ""
Код: 218

Функция FRE(0) подсчитывает количество свободных байтов памяти. Функция FRE"" указывает компьютеру провести так называемую "чистку", которая представляет собой средство очистить память от символьных последовательностей (при этом "чистка" начинается точно под адресом HIMEM и продолжается в направлении убывания адресов). Это полезно, если во время работы программы символьным последовательностям присваиваются новые значения. Если новые последовательности более короткие, компьютер сохраняет тот же размер памяти, дополняя более короткие последовательности пробелами. Если они более длинные, необходимо отвести им новое место, а все первоначально выделенное пространство останется неиспользованным. Фактически функция FRE"" сгущает пространство, занятое символьными последовательностями. Таким образом неиспользуемая память освобождается.

```
10 REM --- FRE ---
20 PRINT FRE(0)
30 A$="A":B$="B"
40 REPEAT
50 A$=A$+A$:B$=B$+B$
60 UNTIL LEN(A$)=128
70 PRINT FRE(0)
80 PRINT "А ТЕПЕРЬ ОСВОБОДИМ ЗАНЯТУЮ ПАМЯТЬ:"
90 A$="":B$=""
100 PRINT FRE("")
```

GRAB

Формат: GRAB
Код: 159

Значительная часть памяти компьютера Правец-8Д отведена для графической страницы в режиме HIRES. Если создается длинная программа на языке БЕЙСИК, в которой не предусмотрена работа в графическом режиме, возможно использование и области памяти, зарезервированной для графики. Это осуществляется с помощью программы GRAB, которая в общем случае вводится в режиме немедленной обработки. Эту команду можно использовать и в теле программы, когда необходима дополнительная память для сохранения массивов.

После активирования команды GRAB графическую страницу для режима HIRES невозможно использовать, пока ее память не будет освобождена (адреса с #9800 до #B400). Освобождение производится командой RELEASE, которая вводится как команда немедленной обработки.

```
10 REM --- GRAB ---
20 CLS
30 PRINT "СВОБОДНАЯ ПАМЯТЬ:"FRE(0)
40 PRINT "БРОНИРОВАНИЕ ПАМЯТИ ПОСРЕДСТВОМ GRAB:"
50 GRAB
60 PRINTFRE(0)
70 PRINT "А ТЕПЕРЬ ВЕРНЕМ ЕЕ ДЛЯ ИСПОЛЬ
ЗОВАНИЯ В РЕЖИМЕ HIRES:"
80 RELEASE
90 PRINTFRE(0)
100 REM
110 REM CM. HIRES, RELEASE
```

RELEASE

Формат: RELEASE
Код: 160

Команда GRAB делает доступной для программы область, предназначенную для графической страницы в режиме HIRES. Подобная необходимость возникает в случае длинных программ, которые не используют режим HIRES. Команда RELEASE возобновляет режим HIRES, вновь отводя байты с #9800 по #B400 для графической страницы.

TRON

Формат: TRON
Код: 132

Устанавливает режим трассировки программы. Команда TRON прослеживает выполнение программы и помогает обнаруживать ошибки. Она выводит на экран номер каждой выполненной строки.

Пара команд TRON и TROFF дает программисту возможность проследить последовательность выполнения программных строк во время прогона программы. В следующем примере задан бесконечный цикл. Команда TRON показывает номера строк в порядке их выполнения.

```
10 REM --- TRON ---
20 CLS
30 TRON
40 FOR A=1 TO 10
50 PRINT A
60 IF A=6 THEN A=1
70 NEXT
80 END
90 REM
99 REM CM. TROFF
```

Если добавить к программе строку

```
65 TROFF
```

то номера строк будут выводиться на экран лишь при первом прохождении цикла FOR...NEXT.

TROFF

Формат: TROFF
Код: 133

Команда TROFF отменяет режим трассировки, установленный командой TRON.

2.9. ОПЕРАЦИИ ВВОДА-ВЫВОДА

INPUT

Формат: INPUT [*s*;] *v1*, *v2*...
Код: 146

Команда INPUT выводит необязательную символьную последовательность *s* на экран и ждет ввода значений переменных *v1*, *v2* и т.д. Практически команда позволяет вводить данные с клавиатуры. Выполнение программы приостанавливается, пока пользователь не введет слово, букву или число. Если, например, ввести команду

```
10 INPUT N$
```

то она остановит программу, после того как изобразит на экране вопросительный знак. Необходимо ввести данные и нажать клавишу RETURN. В нашем случае вероятно лишь программист может сообразить, что именно необходимо ввести. Неплохо поэтому вставить и какое-нибудь сообщение, например:

```
10 PRINT@13,10;"КАК ТЕБЯ ЗОВУТ";
20 INPUT N$
30 REM CM. GET, KEY$
```

или

```
10 INPUT "КАК ТЕБЯ ЗОВУТ";N$
```

Преимущество первого варианта заключается в возможности позиционировать сообщение на произвольное место экрана, тогда как во втором варианте оно выводится на текущую позицию курсора.

GET

Формат: GET *v*
Код: 190

Команда GET останавливает программу и ждет нажатия произвольной клавиши. Значение введенного символа присваивается переменной *v*. Выполнение программы

возобновляется автоматически сразу после нажатия клавиши (в отличие от команды INPUT, где необходимо нажать и RETURN).

Команда GET очень удобна для программ с меню, в которых требуется вводить лишь по одному символу. Известные сообщения "ВЫБЕРИТЕ" или "НАЖМИТЕ ПРОИЗВОЛЬНУЮ КЛАВИШУ, ЧТОБЫ ПРОДОЛЖИТЬ" используют возможности команды GET, например:

```
10 REM --- GET ---
20 HIRES :C=RND(1)*6+1:INK C:PAPER 0
30 FOR A=10 TO 50 STEP 10
40 CURSET 50+(A*2),96,3
50 CIRCLE 10+A,2
60 NEXT
70 PRINT "НАЖМИТЕ КЛАВИШУ"
80 GET A$
90 GOTO 20
100 REM
110 REM СМ. INPUT, KEY$
```

Переменная в команде GET может быть и числовой. Тогда допустимыми символами будут лишь десять цифр на клавиатуре.

Подобно команде GET функция KEY$ также принимает один символ, но, в отличие от GET, KEY$ не ждет, а передает управление следующей строке программы, даже если никакая клавиша не нажата.

KEY$

Формат: *v*$ = KEY$
Код: 241

Функция KEY$ является одним из средств, с помощью которых Правец-8Д получает информацию извне. Эта встроенная функция проверяет, не была ли нажата какая-нибудь клавиша клавиатуры и передает программе соответствующий символ. В отличие от GET компьютер не ждет обязательного ответа с клавиатуры. Выполнение программы не прерывается. Нажатие предварительно определенной клавиши может по существу внести изменение в ход выполнения программы. Если же клавиша не будет нажата, функция KEY$ принимает пустую строку и программа продолжает работу со следующего оператора.

Функция KEY$ очень полезна для некоторых игр, так как передает программе значение символа нажатой клавиши. Таким образом произвольные клавиши могут использоваться для управления "движением" на экране.

```
10 REM --- KEY$ ---
20 CLS:X=2
30 PRINT"'J'-НАЛЕВО, 'K'-НАПРАВО, 'S'-СТОП"
40 REPEAT
50 V$=KEY$
60 IFV$="J"THENX=X-3:PLOTX+3,10,"     "
70 IFV$="K"THENX=X+3:PLOTX-3,10,"     "
80 IF X<2 THEN X=2
90 IF X>35 THEN X=35
100 PLOT X,10,"<*>"
110 UNTIL V$="S"
120 PRINT "КОНЕЦ"
130 REM
131 REM CM. GET, INPUT
```

READ

Формат: READ *v1, v2,...*
Код: 149

Команда READ читает значения для переменных *v1, v2,...* из списка, заданного в командах DATA. READ читает последовательно, начиная со списка первой команды DATA. Если число переменных в операторе READ превышает число элементов в операторах DATA, возникает ошибка OUT OF DATA. Очевидно, что критическим является местоположение операторов READ, в то время как DATA могут находиться в любой точке программы.

```
10 REM --- READ ---
20 CLS
30 FOR A=1 TO 5
40 READ V,V$
50 PLOT V,10,V$
60 NEXT
70 DATA  2,ПРАВЕЦ,8,-,9,8Д,12,МИКРОПРОЦЕССОР,26,6502
80 REM
90 REM CM. DATA, RESTORE
```

DATA

Формат: DATA *n1*, *n2*,...
Код: 145

Команда DATA задает список данных, которые можно читать и присваивать как значения переменных с помощью READ. Элементы списка *n1*, *n2* и т.д. могут быть числами, символами или строковыми данными. Если данные содержат ведущие интервалы, они должны быть заключены в кавычки. Отдельные элементы в списке разделяются интервалами. Если необходимо сделать запятую элементом списка, она заключается в кавычки. Команды DATA могут располагаться в програме произвольно, независимо от места, где элементы их списков читаются командой READ. Элементы списка читаются слева направо. Указатель чтения может быть вновь установлен в начале списка командой RESTORE. Обычно это делается, когда необходимо повторное чтение данных.

В следующей программе данные, заданные командами DATA в строках 70 и 80, читаются и выводятся на экран в строке 30:

```
10 REM --- DATA ---
20 FOR I=1 TO 2: FOR J=1 TO 2: READ A$,A
30 PRINT A$,A
40 NEXT
50 NEXT
60 END
70 DATA "ЗДРАВСТВУЙТЕ",1,"ПРОГРАММИСТЫ",2
80 DATA "        НА     ",3,"ПРАВЕЦ-8Д",4
90 REM
100 REM CM. READ И RESTORE
```

RESTORE

Формат: RESTORE
Код: 154

После того, как некоторый элемент списка, определенного командой DATA, прочитан, специальный указатель перемещается к следующему элементу списка. Когда в рамках одной программы данные из списка нужно читать более одного раза, используется команда RESTORE, которая возвращает указатель в начало списка (в противном случае компьютер выдает сообщение об ошибке OUT OF DATA).

```
10 REM --- RESTORE ---
20 HIRES:PAPER 1:INK 0
30 C=1
40 FOR A=1 TO 5
50 IF C=0 THEN WAIT 20:SHOOT
60 READ V
70 CURSET V,40,3
80 FOR B=1 TO 18 STEP B
90 CIRCLE B,C
100 NEXT B
110 NEXT A
120 IF C=0 THEN END
130 RESTORE
140 C=0
150 GOTO 40
160 DATA 28,73,118,163,208
170 REM
180 REM CM. DATA, READ
```

PRINT

Формат: PRINT *список*
PRINT @ *x, y; список*
Код: 186

Команда PRINT выводит на экран указанные элементы. Список элементов может включать числа, числовые переменные, строковые данные, выражения, которые отделяются друг от друга запятой, точкой с запятой или просто пробелом.

Если PRINT используется самостоятельно, без елементов, компьютер выводит на экран пустую строку.

Вместо PRINT можно использовать вопросительный знак ?.

Использование запятой или точки с запятой как разделителя между отдельными элементами списка управляет их выводом на экран. Разделитель ; сохраняет позицию курсора непосредственно за последним выведенным элементом, так что PRINT продолжит выводить на ту же строку. Вывод будет проводиться таким же образом и без использования разделителя. Разделитель ; просто делает команду более наглядной, а в ряде случаев должен использоваться обязательно для разграничения элементов данного списка.

Если PRINT заканчивается точкой с запятой, то следующая команда PRINT продолжает выводить на ту же строку.

Использование запятой в качестве разделителя между элементами списка перемещает курсор в начало следующего поля экрана. В компьютере Правец-8Д экран разделен (табулирован) на пять полей, каждое с шириной 8 позиций. Запятая передвигает курсор на одно поле вправо и затем к началу следующего поля, обеспечивая тем самым наличие по крайней мере одного интервала между элементами, форматированными таким образом. Использование нескольких запятых, стоящих одна за одной, перемещает курсор на несколько полей вправо.

Команда PRINT позволяет вставлять управляющие символы, используя при этом функцию CHR$(*n*) (где *n* - ASCII код символа). Управляющие символы занимают по одной позиции в карте памяти экрана. На самом экране это отображается одной пустой позицией (интервалом).

В следующем примере демонстрируются различные способы использования команды PRINT:

```
10 REM --- PRINT ---
20 B=2
30 A$="ПРАВЕЦ-8Д"
40 PRINT 1,2,3
50 PRINT A$
60 PRINT "TO BE OR NOT TO BE"
70 PRINT 2*B OR NOT 2*B
80 PRINT
90 PRINT 1;2;3
```

Формат PRINT @*x*,*y* определяет координаты (*x*,*y*) позиции экрана в режиме TEXT или LORES, с которой начинается вывод элементов. Таким образом можно управлять текущей позицией курсора.

Координата *x* является номером столбца, а *y* - номером строки. Значения *x* и *y* разделяются запятой, за ними следует точка с запятой, отделяющая их от списка элементов, на который распространяются те же правила, что и для обычного формата PRINT. Координаты *x* и *y* могут быть переменными или числовыми выражениями. Если они не являются целыми числами, то округляются. Если же их значения оказываются вне допустимых границ, компьютер выдает сообщение ILLEGAL QUANTITY.

Следующая программа вычерчивает окружность, используя функции SIN и COS, чтобы вычислить позиции X и Y в PRINT@:

```
10 REM --- PRINT @ ---
20 R=13:XC=20:YC=13:CLS
30 FOR A=0 TO 2*PI STEP PI/50
40 X=XC+COS(A)*R
50 Y=YC+SIN(A)*R
60 PRINT @X,Y;"*"
70 NEXT
80 REM
90 REM CM. LPRINT,SPC,TAB
```

LPRINT

Формат: LPRINT
Код: 143

Команда аналогична команде PRINT, но вместо на экран выводит информацию на печатающее устройство. Недопустим формат LPRINT@. В комбинации с управляющими кодами LPRINT может управлять выводом на различные печатающие устройства. Длина строки может изменяться командой POKE #256,*n*, где *n* - длина строки в символах.

TEXT

Формат: TEXT
Код: 161

Команда TEXT возвращает компьютер в стандартный текстовый режим (как при начальном включении) с 27 строками, 40 столбцами и стандартным набором символов. Использование команды LORES или HIRES активирует графические режимы экрана, которые остаются до тех пор, пока не будут отменены соответственно с помощью CLS или TEXT.

CLS

Формат: CLS
Код: 148

Эта команда очищает экран, заполняет его цветом фоновой области и позиционирует курсор в левом верхнем углу. Желательно все программы с текстом начинать с

команды CLS, если только не существует основательных причин оставить предыдущий текст на экране. В примере экран очищается два раза: первый раз в строке 20, когда он подготавливается для вывода и второй раз в строке 70, после того как текст заполнил экран (происходит то же самое, что и при использовании CTRL-L).

```
10 REM --- CLS ---
20 PAPER 0 : INK 5 : CLS
30 FOR A=0 TO 134
40 PRINT "ПРАВЕЦ-8Д",
50 NEXT
60 WAIT 100
70 CLS
80 WAIT 100
90 GOTO 20
100 REM
110 REM CM. HIRES, LORES И TEXT
```

SPC

Формат: SPC (*n*)
Код: 197

Функция SPC выводит на экран *n* пробелов. Если *n* не является целым числом, то оно округляется книзу. Можно использовать также выражения, а возможность и использования внутри скобок функций, которые работают с цепочками символов, делает SPC очень полезной функцией при форматировании. Она используется в командах PRINT, чтобы задать требуемое количество пробелов до или между отдельными элементами PRINT, например:

```
10 REM --- SPC ---
20 FOR F=0 TO 20
30 PRINT "*";SPC(F);"*";F;"ПРОБЕЛОВ"
40 NEXT
```

```
** 0 ПРОБЕЛОВ
* * 1 ПРОБЕЛОВ
*  * 2 ПРОБЕЛОВ
*   * 3 ПРОБЕЛОВ
*    * 4 ПРОБЕЛОВ
*     * 5 ПРОБЕЛОВ
*      * 6 ПРОБЕЛОВ
*       * 7 ПРОБЕЛОВ
```

```
*          * 8 ПРОБЕЛОВ
*           * 9 ПРОБЕЛОВ
*            * 10 ПРОБЕЛОВ
*             * 11 ПРОБЕЛОВ
*              * 12 ПРОБЕЛОВ
*               * 13 ПРОБЕЛОВ
*                * 14 ПРОБЕЛОВ
*                 * 15 ПРОБЕЛОВ
*                  * 16 ПРОБЕЛОВ
*                   * 17 ПРОБЕЛОВ
*                    * 18 ПРОБЕЛОВ
*                     * 19 ПРОБЕЛОВ
*                      * 20 ПРОБЕЛОВ
```

Эта программа использует цикл для определения числа пробелов между звездочками и указывает при этом сколько пробелов она оставила. Функция SPC полезна как альтернатива функции TAB, гибкость ее иллюстрируется следующим примером. Значения, полученные в цикле FOR..NEXT, превращаются в символьную последовательность с использованием STR$ в строке 40, а затем в строке 50 с использованием функции LEN вычисляется количество пробелов. Таким образом вывод на экран будет симметричным относительно его середины.

```
10 REM --- SPC'2 ---
20 FOR F=1 TO 19
30 N%=F^2*47
40 N$=STR$(N%)
50 PRINT SPC(20-LEN(N$));N%
60 NEXT
70 REM
80 REM CM. PRINT, LPRINT, TAB
```

TAB

Формат: TAB (*n*)
Код: 194

Функция TAB используется командой PRINT для позиционирования ее элементов на определенном столбце экрана. Значение аргумента *n* определяет столбец, в который переместится позиция для PRINT. Очередной элемент, подлежащий выводу, будет отображен начиная с этой позиции, если за функцией TAB(*n*) нет разделителя или имеется точка с запятой. Если после TAB стоит запятая, очередной элемент

будет выведен в следующее стандартное поле. Если числа не отрицательные, они отображаются с ведущими пробелами. Столбцы экрана пронумерованы от 0 до 39. Следующая программа иллюстрирует эффект защищенных столбцов:

```
10 REM --- TAB ---
20 FOR F=0 TO 15
30 PRINT TAB (F);F
40 NEXT
50 PRINT "ВВЕДИТЕ CTRL-J И ПОПРОБУЙТЕ СНОВА!"
```

В одну команду PRINT можно включить несколько функций TAB. Это показано в следующем примере:

```
10 REM ---TAB'2 ---
20 PRINT TAB(10)10;TAB(20);20
30 PRINT TAB(10);10;TAB(20),-20
40 PRINT TAB(10)"X"TAB(20),"Y"
50 REM
60 REM СМ. LPRINT, PRINT, POS, SPC
```

POS

Форматы: POS (0)
POS (1)
Код: 219

Функция POS определяет текущую горизонтальную позицию курсора, полученную от последней команды PRINT. Ее невозможно использовать для позиций, определенных посредством PLOT или PRINT@. Формат POS(0) дает позицию курсора на экране, а POS (1) определяет горизонтальную позицию печатающей головки при выполнении команды LPRINT.

```
10 REM --- POS ---
20 DIM A(4,2)
40 FOR K=1 TO 4
50 FOR J=1 TO 2
60 A(K,J)=K*RND(1)+]
70 NEXT
80 NEXT:CLS
90 FOR K=1 TO 3:PRINT:NEXT
100 PRINTA(1,1);
110 REPEAT:PRINT"  ";:UNTIL POS(0)=23
120 PRINT A(1,2)
129 REM
130 REM СМ. PRINT
```

2.10. ГРАФИКА

LORES

Формат: LORES 0
LORES 1
Код: 137

Эта команда устанавливает графический режим с низкой разрешающей способностью и обеспечивает два из четырех режимов экрана Правец-8Д (остальные два режима TEXT и HIRES). Формат экрана - 27 строк по 40 символов.

Команда LORES 0 обеспечивает экран, подобный экрану в режиме TEXT, но генерирует черный фон, на котором изображаются светлые (белые) символы. Сами символы должны позиционироваться с помощью PRINT @ или PLOT, так как простое использование PRINT переключает экран в текстовый режим.

Команда LORES 1 действует абсолютно таким же образом, с той разницей,что изображает на экране альтернативный набор символов. Следующая программа демонстрирует использование обоих режимов экрана:

```
10 REM --- LORES ---
20 LORES 0:C=0
30 PLOT 2,24,"СИМВОЛЬНЫЙ НАБОР В LORES 0"
40 FOR A=32 TO 126
50 X=RND(1)*36+1:Y=RND(1)*22+1
60 C$=CHR$(A)
70 PLOT X,Y,C$
80 WAIT 50
85 NEXT
90 IF C=1 THEN WAIT 500:CLS:END
100 PRINT:PRINT"А ТЕПЕРЬ В LORES 1":WAIT 300
110 GOSUB 130
120 GOTO 30
130 CLS:LORES1:C=1
140 RETURN
149 REM
150 REM СМ. TEXT, HIRES
```

PAPER

Формат: PAPER *n*
Код: 177

Эта команда определяет цвет фона экрана. Она действует в режимах низкой и высокой разрешающей способности. В параметре *n* команды указывается код одного из следующих цветов:

0 - черный;
1 - красный;
2 - зеленый;
3 - желтый;
4 - синий;
5 - лиловый;
6 - светло-синий;
7 - белый.

Чтобы определить различный фон для отдельных частей экрана, используются CHR$ и FILL.

```
10 REM --- PAPER ---
20 HIRES:INK0
30 A=0:B=0
40 FOR V=0 TO 25
50 A=A+1:B=B+3
60 IF A>7 THEN A=0
70 PAPER A: WAIT 20
80 CURSET 110,100,3
90 CIRCLE B,1
100 NEXT V
110 GOTO 20
120 REM
130 REM CM. LORES, HIRES, INK, TEXT, CHR$
```

INK

Формат: INK *n*
Код: 178

Эта команда работает в режимах высокой и низкой разрешающей способности. Она определяет цвет всего, что пишется на экране. Выбор цвета определяется параметром *n*. Компьютер Правец-8Д предлагает 8 цветов:

0 - черный;
1 - красный;
2 - зеленый;
3 - желтый;
4 - синий;

5 - лиловый;
6 - светло-синий;
7 - белый.

Команда INK изменяет цвет "чернил", которыми пишется что выводится на экран, и не может использоваться для отдельных символов в различном цвете.

```
10 REM --- INK ---
20 HIRES
30 A=0:B=0
40 FOR V=0 TO 25
50 B=B+1:A=A+3
60 IF B>7 THEN B=0
70 INK B
80 CURSET 110,100,3
90 CIRCLE A,1
100 NEXT V
110 GOTO 20
120 REM
130 REM CM. LORES, HIRES, PAPER, TEXT, CHR$
```

PLOT

Формат: PLOT *x*, *y*, *n*
PLOT *x*, *y*, *s*
Код: 135

Эта команда используется для позиционирования символов на экране с низкой разрешающей способностью.

В первом формате команды *n* должно быть числовым выражением. Выражение дает ASCII код, а на экран выводится соответствующий символ. Могут использоваться стандартный и альтернативный наборы символов, что позволяет создавать на экране интересные изображения. Например,

```
10 PLOT 11,11,12:PLOT 15,11,"ПРАВЕЦ-8Д"
```

выведет на экран мигающую надпись Правец-8Д. Практика раскроет возможности команды, используемой таким образом. Формат PLOT *x,y,s* выводит строку *s* на экран, начиная с позиции (*x,y*). Координаты как для числовых, так и для символьных выражений изменяются между 0-39 для *x* и 0-26 для *y*. Команда работает в режимах TEXT, LORES0 и LORES1. Она недопустима в режиме HIRES.

```
10 REM --- PLOT ---
20 CLS:LORES 0
30 PLOT 2,2,"PLOT В РЕЖИМЕ LORES 0"
40 FOR A=1 TO 23
50 X=RND(1)*31+4:Y=RND(1)*22+3
60 PLOTX,Y,A:PLOT X+2,Y,"ПРАВЕЦ-8Д"
70 WAIT 50:NEXT
80 REM
90 REM СМ. CHAR, PRINT
```

SCRN

Формат: SCRN(*x*, *y*)
Код: 242

Функция SCRN определяет ASCII код символа в избранной точке экрана (заданной координатами *x* и *y*). Она используется только в режимах LORES и TEXT. Для *x* и *y* можно задавать числовые выражения, значения которых должны быть в интервале 0-39 для *x* и 0-26 для *y*, в противном случае последует сообщение об ошибке ILLEGAL QUANTITY.

Приведенные ниже простые примеры иллюстрируют возможности SCRN. В первом используется PRINT@, а во втором - PLOT. Знак # изображается на экране в позиции (5,5) двумя способами. Строка 30 использует SCRN сначала, чтобы вывести код ASCII знака #, а затем то же самое выражение SCRN используется с функцией CHR$. Оно выводит в позицию, определенную координатами x и *y*, действительный символ, найденный в памяти.

```
10 CLS
20 PRINT @5,5;"#"
30 PRINT SCRN (5,5),CHR$(SCRN(5,5))
```

```
10 CLS
20 A$="#"
30 PLOT 5,5,A$
40 PRINT SCRN(5,5),CHR$(SCRN(5,5))
```

Функция SCRN работает также с повторно определенными символами. В следующем примере звездочка используется как ракета, которая запускается вверх и движение которой влево и вправо управляется соответствующими клавишами для движения курсора. Повторно определенный символ ! оформляет появляющиеся на экране цели. Цель поражается, когда SCRN (x,y) получит значение 33 - код знака !.

```
10 REM --- SCRN ---
20 CLS:FOR I=1 TO 8
30 READ A:POKE 46343+I,A
40 NEXT:POKE #24E,1:POKE #24F,1
50 PRINT @2,0;"СТРЕЛЬБА - ПРОБЕЛ; <- И -> ДВИЖЕНИЕ"
60 FOR I=1 TO 10:PRINT @INT(RND(1)*37)+2,1;"!";
70 NEXT:PRINT CHR$(17);CHR$(6)
80 REPEAT
90 REPEAT:K$=KEY$:UNTIL K$=" " OR K$=
CHR$(13):IF K$=CHR$(13) THEN 220
100 X=20:Y=26
110 PRINT @X,Y;"*";:SHOOT
120 REPEAT:OX=X
130 IF KEY$=CHR$(8) AND X>2 THEN X=X-1
140 WAIT 1
150 IF KEY$=CHR$(9) AND X<39 THEN X=X+1
160 Y=Y-1
170 PRINT @OX,Y+1;" ";
180 IF SCRN(X,Y)=33 THEN PRINT @X,Y;"
":EXPLODE:Y=1:GOTO 200
190 PRINT @X,Y;"*"
200 UNTIL Y=1:PRINT@X,Y;" "
210 UNTIL KEY$=CHR$(13)
220 POKE #24E,32:POKE #24F,4:PRINT CHR$(17);CHR$(6)
230 END
300 DATA 30,33,45,63,45,33,30,0
399 REM
400 REM CM. ASC, PLOT, PRINT @
```

HIRES

Формат: HIRES
Код: 162

Команда устанавливает режим работы с высокой разрешающей способностью (горизонтально 240 точек и вертикально 200 точек). В этом режиме экрана максимально раскрываются графические возможности компьютера. Правец-8Д предлагает серию специализированных команд, которые работают в этом режиме: CURSET, CURMOV, DRAW, FILL, CIRCLE, PATTERN, CHAR и другие.

Команда подается в двух других основных режимах - текстовом (TEXT) и графическом с низкой разрешающей способностью (LORES). Первые 24 строки экрана изменяются в

фоновой области на черный цвет. Три нижние строки остаются в нормальном режиме TEXT и используются для написания команд и сообщений.

Естественно, HIRES занимает значительную часть памяти компьютера. Эта зарезервированная память может использоваться программами на языке БЕЙСИК с помощью команды GRAB. Команда RELEASE вновь возвращает память для работы в режиме HIRES.

Чтобы рассмотреть операторы программы в режиме HIRES, сначала необходимо перейти в режим TEXT. Затем можно использовать LIST и, если необходимо, отредактировать программу.

В следующем примере команды до FILL включительно выполняются только в режиме HIRES.

```
5 REM --- HIRES ---
10 HIRES
20 CURSET 14,10,3
30 DRAW 0,180,1:DRAW 220,0,1
40 CURMOV -210,-10,3:DRAW -19,19,1
50 FOR B=1 TO 6
60 READ H:GOSUB 100
70 NEXT B
80 GOSUB 400:END
90 REM РИСОВАНИЕ СТОЛБЦОВ
100 X=(B*5+2)*6-5:CURSET X,190,3
110 DRAW 0,-H,1:DRAW 10,-10,1:DRAW 12
,0,1:DRAW -10,10,1
120 CURSET X+23,180-H,3
130 FILL H,1,16
140 CURSET X,190-H,3
150 FILL H,2,B+16
160 CURSET X+13,190-H,3
170 FILL H,1,16
200 RETURN
300 DATA 100,75,90,124,150,170
400 REM ВЫВОД ТЕКСТА ПОСРЕДСТВОМ CHAR
410 A$="ПРОДАЖА КОМПЬЮТЕРОВ ПРАВЕЦ-8Д"
420 FOR I=1 TO LEN(A$)
430 CURSET I*6+40,10,3
440 CHAR ASC(MID$(A$,I)),0,1
450 NEXT
460 PATTERN 51:CURSET 40,20,3
470 DRAW 135,0,1
480 RETURN
500 REM
501 'CM. CHAR, CIRCLE, CURMOV, CURSET,
```

```
502 ' DRAW, FILL, LORES, TEXT, PAPER, INK
```

Далее следует группа специализированных графических команд для режима HIRES. Большинство из них имеют последний операнд f, который в зависимости от своего значения имеет следующие функции:

f=0 - устанавливает фоновый цвет (тот, который определен командой PAPER);

f=1 - устанавливает основной цвет (определенный командой INK);

f=2 - заменяет фоновый цвет и основной цвет;

f=3 - не выводит на экран.

CURSET

Формат: CURSET *x*,*y*,*f*
Код: 170

Эта команда устанавливает курсор в заданной позиции экрана в режиме HIRES, при этом *x* и *y* являются абсолютными координатами в рамках экрана (*x* принимает значения от 0 до 239, а *y* от 0 до 199). Как всегда, *f* является целым числом от 0 до 3 (см. HIRES). Так как команда CIRCLE позиционирует центр окружности в текущей позиции курсора, следующий пример является хорошей демонстрацией действия CURSET:

```
10 REM --- CURSET ---
20 HIRES : INK 5 : PAPER 4
30 FOR C=0 TO 100 STEP 20
40 FOR B=1 TO 0 STEP -1
50 CURSET 20,50+C,0
60 FOR A=0 TO 30
70 CIRCLE 10+A,B
80 CURMOV 5,0,0
90 NEXT A
100 NEXT B
110 NEXT C
120 REM
130 REM CM. CIRCLE, CURMOV, DRAW, HIRES
```

CURMOV

Формат: CURMOV *x*,*y*,*f*
Код: 171

Эта команда используется только в режиме HIRES и перемещает курсор в новую позицию. Значения *x* и *y* задаются относительно текущей позиции курсора. Параметр *f* принимает значения 0, 1, 2 или 3 (см. HIRES). Команда полезна при изображении движения и используется в большинстве графических процедур. В примере, приведенном для CURSET, команда CURMOV перемещает окружность по экрану и затем стирает ее, используя соответствующее значение *f*.

PATTERN

Формат: PATTERN *n*
Код: 174

PATTERN (шаблон) является командой, которая работает только в режиме высокой разрешающей способности. Используется в комбинации с DRAW или CIRCLE.

Нормально DRAW и CIRCLE изображают непрерывную линию. Команда PATTERN превращает ее в линию из точек или тире, при этом используемый шаблон определяется заданным в команде PATTERN целым числом *n*, лежащим в интервале 0-255. Следующая программа демонстрирует полные возможности PATTERN.

```
10 REM --- PATTERN ---
20 HIRES:INK 0:PAPER 1
30 FOR A=1 TO 65
40 PATTERN 170-A
50 DRAW 230,0,1
60 CURMOV -230,3,0
70 NEXT
80 REM
90 REM СМ. HIRES, DRAW, CIRCLE
```

DRAW

Формат: DRAW *x,y,f*
Код: 172

Команду DRAW можно использовать только в режиме HIRES. Она работает в комбинации с CURSET и CURMOV и вычерчивает непрерывную линию от текущей позиции курсора до точки с координатами *x* и *y*. Указанные в команде значения *x* и *y* не являются абсолютными координатами, а

задают смещение относительно текущей позиции курсора. Значение *x* должно быть в интервале 0-199, а *y* - в интервале 0-239. Выход за рамки экрана вызывает сообщение об ошибке. Как обычно, параметр *f* принимает значения от 0 до 3.

Комбинация DRAW с PATTERN превращает непрерывную линию в пунктирную или штриховую определенной густоты (см. PATTERN).

Ниже следует интересная демонстрация команды DRAW:

```
10 REM --- DRAW ---
20 HIRES:PAPER 6:INK 1
30 FOR A=10 TO 230 STEP 10
40 CURSET A,0,0
50 DRAW 0,199,1
60 WAIT 20:PING
70 NEXT
80 FOR Y=10 TO 190 STEP 10
90 CURSET 0,Y,0
100 DRAW 239,0,1
110 WAIT 20:PING
120 NEXT
130 REM
140 REM CM. CIRCLE, CURMOV, CURSET,
150 REM     HIRES И PATTERN.
```

CIRCLE

Формат: CIRCLE *r*,*f*
Код: 173

В режиме высокой разрешающей способности эта команда вычерчивает окружность с центром в текущей позиции курсора. Обычно CIRCLE используется в комбинации с командой CURSET. Если часть окружности окажется вне экрана, компьютер выдает сообщение ошибке. Длина радиуса *r* задается числом точек (1-99). В программе вычерчивается несколько окружностей, центры которых расположены в различных точках экрана. Затем изображение стирается посредством *f*=0 (фоновый цвет) (см. HIRES).

```
10 REM --- CURCLE ---
20 HIRES
30 FOR A=0 TO 60 STEP 5
40 FOR B=1 TO 0 STEP -1
50 CURSET 80+A,100,0
```

```
60 CIRCLE A+9,B
70 NEXT B
80 NEXT A
90 END
100 REM
110 REM CM. CURMOV, CURSET, DRAW, HIRES
120 REM     И PATTERN
```

CHAR

Формат: CHAR *n*,*m*,*f*
Код: 176

Команда CHAR пишет символ с ASCII кодом, равным *n*, из набора символов, определенного *m* (0 для стандартного или 1 для альтернативного набора), располагая при этом верхний левый конец символа в текущей точке экрана в режиме HIRES.

Аналогичными командами для режима LORES являются PRINT, PLOT и INPUT.

Команда особенно полезна для графических изображений и игр. Она не перемещает курсор, но в комбинации с CURMOV и CURSET дает возможность позиционировать текст или символы пользователя с точностью до одной точки.

```
10 REM --- CHAR ---
20 HIRES : C=0
30 FOR A=0 TO 150 STEP 50
40 CURSET 40+A,90,3
50 DRAW 20,0,2 :DRAW 0,20,2
60 DRAW-20,0,2 :DRAW 0,-20,2
70 CURMOV 7,6,3
80 CHAR 49+C,0,2
90 C=C+1
100 NEXT
110 REM
120 REM CM. CURMOV, CURSET, HIRES, PLOT
```

FILL

Формат: FILL *x*,*y*,*n*
Код: 175

Команда FILL используется только в режиме HIRES. Она дает возможность заполнения блоками из 6 точек экрана. Экран в режиме HIRES имеет 200 строк по 240 точек в строке. Таким образом строка для FILL содержит 40 блоков.

Команда заполняет *y* строк по *x* байтов значением *n* - двоичным шаблоном для каждого байта. Поэтому *y* лежит в интервале от 1 до 199, *x* - от 1 до 40, а *n* должно быть между 0 и 255.

Команду FILL можно использовать, чтобы раскрасить отдельные части экрана компьютера в различные цвета. Она удобна для заполнения небольших фигур неправильной формы, очень быстро заполняет прямоугольные фигуры. За начало описанного блока (верхний левый угол) принимается позиция курсора. После заполнения блока курсор остается в его нижнем левом углу. Следующая программа демонстрирует действие FILL.

```
10 REM --- FILL ---
20 PAPER 0
30 HIRES:PRINT CHR$(17)
40 REPEAT
50 X=INT(RND(1)*231)
60 Y=INT(RND(1)*175+7)
70 A=INT(RND(1)*(230-X)/6+1)
80 D=INT(RND(1)*(182-Y)+8)
90 CURSET X+6,Y-7,0:FILL D+7,A,16
100 CURSET X,Y,0:FILL D,A,(17+RND(1)*7)
110 UNTIL FALSE
120 REM
130 REM CM. CURSET, CURMOV, HIRES
```

POINT

Формат: POINT (*x*,*y*)
Код: 243

Функция проверяет цвет экрана в режиме HIRES в точке с координатами (*x*,*y*) и имеет значение 0, если цвет выбран командой PAPER, и значение 1, если выбран командой INK. Допустимые значения для *x* от 0 до 239, а для *y* от 0 до 199.

Функция POINT часто используется в игровых программах для обнаружения столкновения двух объектов. Следующая программа предлагает простую демонстрацию:

```
10 REM --- POINT ---
20 HIRES:PAPER 4:INK 0
30 FOR B=5 TO 175
40 CURSET 200,B,2:CHAR 124,0,2
50 NEXT B
```

```
60 A=5
70 REPEAT
80 A=A+8
90 CURSET A,90,0
100 CHAR 127,0,1:WAIT 5
110 CHAR 127,0,0
120 C=POINT(A+11,90)
130 UNTIL C=-1
140 EXPLODE
149 REM
150 REM CM. HIRES
```

2.11. МУЗЫКАЛЬНЫЕ ВОЗМОЖНОСТИ

В отличие от Правец-82, домашний компьютер Правец-8Д обладает специализированным трехканальным музыкальным процессором. Он предлагает 3 специализированные музыкальные команды (SOUND, MUSIC и PLAY), которые позволяют генерировать удивительно широкую гамму музыкальных звуков. Еще четыре команды воспроизводят встроенные звуковые эффекты (ZAP, SHOOT, PING, EXPLODE).

ZAP

Формат: ZAP
Код: 165

Эта команда издает встроенный звук, который имитирует выстрел из некоего оружия будущего, и используется для игр. В отличие от других встроенных звуков (PING, SHOOT и EXPLODE), ZAP не требует использования WAIT при управлении звуком и можно подавать ее многократно.

```
10 REM --- ZAP ---
20 FOR F=1 TO 7
30 PAPER F
40 ZAP
50 NEXT
98 REM
99 REM CM. EXPLODE, SHOOT, PING
```

SHOOT

Формат: SHOOT
Код: 163

Команда имитирует выстрел из ружья. Если используется несколько последовательных SHOOT, то они должны быть разделены командами WAIT, чтобы звук был чистым и отдельные выстрелы не накладывались друг на друга. Если выполнить программу

```
10 REM --- SHOOT ---
20 FOR C=1 TO 6
30 SHOOT
50 NEXT
```

то слышен будет тот же звук, что и при выполнении единственной команды SHOOT, так как цикл заканчивается прежде, чем затихнет звук первого выстрела. Чтобы слышать 6 отдельных выстрелов, необходимо прибавить WAIT:

```
10 REM --- SHOOT'2 ---
20 FOR C=1 TO 6
30 SHOOT
40 WAIT 30
50 NEXT
60 REM
70 REM CM. EXPLODE, PING, ZAP
```

PING

Формат: PING
Код: 166

Команда имитирует звук колокольчика, который можно использовать как сигнал или в игровых программах. В комбинации с WAIT эта команда генерирует эффектную последовательность звуков. Действие команды можно имитировать вводом CTR-G (тот же эффект возникает при попытке ввести в одну программную строку больше 80 символов.

```
10 REM --- PING ---
20 PAPER 1:INK 0:CLS:PRINT
```

```
30 PRINT CHR$(140)"ВЫБЕРИ:"
40 PRINT:PRINT:PRINT
50 PRINT CHR$(147)"1 КОТЕНОК":PING
60 WAIT 30
70 PRINT CHR$(148)"2 ХОМЯК":PING
80 WAIT 30
90 PRINT CHR$(146)"3 ЦЫПЛЕНОК":PING
100 REM
110 REM СМ. EXPLODE, SHOOT, ZAP
```

EXPLODE

Формат: EXPLODE
Код: 164

При выполнении команды раздается звук, имитирующий взрыв. Используется чаще всего в играх, но может быть полезна и в ряде серьезных программ. Для многократного повторения взрывов необходимо использовать паузы (посредством команды WAIT).

```
10 REM --- EXPLODE ---
20 HIRES:PAPER 3:INK 4
30 FOR B=0  TO 95 STEP 4
40 CURSET 120,2+B,0
50 GOSUB 200
60 CURSET 120,190-B,0
70 GOSUB 200
80 NEXT
90 EXPLODE
100 FOR K=1 TO 10
110 PAPER 4:INK 3
120 WAIT K
130 PAPER 3:INK 4
140 WAIT K
150 NEXT
160 WAIT 200
170 GOTO 20
180 END
200 FOR I=1 TO 3
210 CHAR 100,1,1:CURMOV 6,0,0
220 NEXT
230 RETURN
240 REM
250 REM СМ. PING, SHOOT, ZAP
```

MUSIC

Формат: MUSIC *c,o,n,v*
Код: 168

Команда типична для современных домашних компьютеров и используется для генерации музыкального выхода. В ней задаются следующие четыре параметра:

c - канал (от 1 до 3);
o - октава (от 0 до 7);
n - нота (от 1 до 12);
v - сила звука (от 0 до 15).

Это означает, что генерируется нота *n* октавы *o* по каналу *c* с силой *v*. Четыре указанных значения должны быть разделены запятыми. Возможности команды очень широкие, но ее использование является не таким простым делом. Обычно MUSIC комбинируется с командой PLAY, оформляющей сам звук и управляющей количеством действующих звуковых каналов.

Далее приведен небольшой пример. В конце программы используется команда PLAY 0,0,0,0 чтобы выключить звук.

```
10 REM --- MUSIC ---
20 FOR O=0 TO 6
30 FOR N=1 TO 12
40 MUSIC 1,O,N,10
50 PLAY 3,0,7,0
60 WAIT 30
70 NEXT N
80 NEXT O
90 PLAY 0,0,0,0
100 EXPLODE
109 REM
110 REM CM. PLAY, SOUND
```

PLAY

Формат: PLAY *t,s,e,d*
Код: 169

PLAY является одной из сложных звуковых команд, которые предлагают возможности для звука и музыки, намного превосходящие стандартные средства персональных

компьютеров. Естественно, команды PLAY и SOUND требуют в дополнение к желанию использовать их при программировании и чисто музыкальных знаний и практических навыков.

Компьютер Правец-8Д снабжен тремя звуковыми каналами, при этом PLAY является командой, определяющей комбинацию этих каналов. Эти три канала могут одновременно издавать три различных звука.

Формат команды позволяет задавать следующие параметры:

t - выходные каналы для тона (от 0 до 7);
s - шумовые каналы для звука (от 0 до 7);
c - тон моделирования звука (от 0 до 7);
d - продолжительность (от 0 до 32767).

Эффект различных комбинаций каналов становится ясен после непродолжительных упражнений. Следующая таблица упрощает использование команды. В левой колонке даны возможные значения *t* или *s*, а в правой - каналы, которые будут активны при этих значениях:

t или *s*	комбинация каналов
0	все каналы выключены
1	включен канал 1
2	включен канал 2
3	включены каналы 1 и 2
4	включен канал 3
5	включены каналы 1 и 3
6	включены каналы 2 и 3
7	включены все каналы - 1, 2 и 3

Третий параметр команды - е является наиболее трудным для понимания. Он представляет собой целое число, моделирующее "форму" звука, создаваемого компьютером. Им определяется ритм изменения силы звука. Имеется, например, специфичный формат для резкого повышения силы (быстрая атака) и медленного затихания звука. Другой формат начинается сразу с сильного звука, который постепенно затихает. Всего имеется семь форматов (от 1 до 7), которые различным образом моделируют звук. Если переменная е имеет значение 1 или 2, будет моделироваться звук фиксированной длины, а значения от 3 до 7 генерируют продолжительный звук различного типа. Однажды заданный командой PLAY, формат е определяет все следующие звуки. Последний параметр *d* (0-32767) определяет продолжительность звука в тысячных долях секунды (в отличие от WAIT, в которой задаются сотые доли секунды).

Чтобы выключить звуковые каналы после использования MUSIC или SOUND, необходимо выполнить команду PLAY с нулевыми параметрами (PLAY 0,0,0,0).

В следующей программе демонстрируется выполнение команды PLAY с различными комбинациями параметров.

```
10 REM --- PLAY ---
20 CLS:PLAY 0,0,0,0
30 CLS:PRINT:PRINT
40 INPUT"ВВЕДИТЕ ЦИФРУ (0-7) ДЛЯ МОДЕЛИРОВАНИЯ";E
50 IF E<0 OR E>7 THEN 40
60 INPUT"ВЫБЕРИТЕ ПРОДОЛЖИТЕЛЬНОСТЬ (1-65535)";D
70 PRINT"ЭТО P L A Y  1, 0,"E","D
80 SOUND 1,500,0
90 PLAY 1,0,E,D
100 PRINT "НАЖМИТЕ КЛАВИШУ ДЛЯ ВЫКЛЮЧЕНИЯ ЗВУКА"
110 GET A$:GOTO 20
120 REM
130 REM CM. MUSIC, SOUND
```

SOUND

Формат: SOUND *c,p,v*
Код: 167

Компьютер Правец-8Д имеет специализированный звуковой чип. Команда SOUND является третьей настоящей музыкальной командой, подобной MUSIC и PLAY. Она активирует первый, второй или третий канал и генерирует по нему чистый звук (соответственно при *c*= 1,2 или 3) или шум (при *c*= 4,5 или 6) с периодом, определенным параметром *p*, и силой, определенной параметром *v*.

Указанный период *p* управляет тоном созданного звука. Допустимы значения от 0 до 65525, но полезный интервал *p* лежит в пределах от 0 до 4000. Это демонстрируется следующей программой:

```
0 PLAY 1,0,0,0
20 FOR P=0 TO 32767
30 SOUND 1,P,8
40 NEXT
```

Сила звука *v* начинается с 0 (что активирует параметр *e* (моделирование звука) команды PLAY), проходит через 1 (тихий) и достигает 15 (сильный). Обычно используются значения от 3 до 5. Этот параметр не действует без наличия

соответствующей команды PLAY. Команду SOUND можно использовать без PLAY для активирования каналов - в этом случае работает канал для тона 1.

Следующие несколько примеров иллюстрируют гибкость команды SOUND. Независимо от того, что основная структура программ одна и та же, они создают сильно различающиеся звуки.

В первой программе имеется команда PLAY, которая активирует первый канал шума. За ней следуют команды SOUND, разделенные командами WAIT. Канал выбирается параметром 4 (что соответствует первому шумовому каналу), период изменяется в зависимости от значения переменной цикла *p*, а для силы звука используются два различных параметра:

```
10 REM --- SOUND'2 ---
20 FOR P=1 TO 3
30 PLAY 0,1,1,250
40 SOUND 4,8+P,10
50 WAIT 5
60 SOUND 4,3+P,6
70 WAIT 20
80 PLAY 0,0,0,0
90 NEXT P
100 GOTO 20
110 REM
120 REM CM. MUSIC, PLAY, WAIT
```

Параметр моделирования звука команды PLAY действует, когда сила звука в SOUND равна нулю. Если она будет изменена в строках 40 и 60 нашего примера (соответственно SOUND 4,8+P,0 и SOUND 4,5+P,0), то услышите различное звучание при активированном значении е. (Это остается в силе и для следующих двух примеров).

Во второй программе изменен только канал звука в PLAY и SOUND - выбран канал звука 1.

```
10 REM --- SOUND'3 ---
20 FOR P=1 TO 3
30 PLAY 1,0,1,250
40 SOUND 1,8+P,10
50 WAIT 5
60 SOUND 1,3+P,6
70 WAIT 20
80 PLAY 0,0,0,0
90 NEXT P
100 GOTO 20
```

```
110 REM
120 REM CM. MUSIC, PLAY, WAIT
```

В последнем примере устранена команда PLAY из строки 30. Таким образом SOUND используется без команды PLAY, хотя в конце используется всё же PLAY 0,0,0,0, чтобы выключить звук.

```
10 REM --- SOUND'4 ---
20 FOR P=1 TO 3
40 SOUND 1,8+P,10
50 WAIT 5
60 SOUND 1,3+P,6
70 WAIT 20
80 PLAY 0,0,0,0
90 NEXT P
100 GOTO 20
110 REM
120 REM CM. MUSIC, PLAY, WAIT
```

СООБЩЕНИЯ ОБ ОШИБКАХ

Современный компьютер имеет развитую систему диагностики. Во время выполнения программы интерпретатор выдает различные сообщения.

Часть из них поясняет работу программы, другие сообщают об ошибках - например, в синтаксисе отдельных команд. Интерпретатор обнаруживает также несоблюдение технических и программных ограничений компьютера и выдает соответствующие сообщения.

Наиболее трудно распознаются логические ошибки - при их наличии программа выполняется, но не дает ожидаемых результатов.

Сообщение об ошибке содержит информацию об ее характере и номер строки, где она обнаружена.

Далее следуют сообщения об ошибках, выдаваемые копьютером Правец-8Д. Сообщения упорядочены по алфавиту и содержат краткий анализ причин их возникновения.

?BAD SUBSCRIPT ERROR

Сообщение выдается при попытке использовать несуществующий элемент массива. Для определенных по умолчанию массивов это означает, что индексы элемента больше 10. Когда массив определен командой DIM, сообщение показывает, что индексы вышли за объявленные границы.

?BAD UNTIL ERROR

Ошибка появляется при попытке выполнить команду UNTIL, перед которой не выполнена соответствующая команда REPEAT.

?CAN'T CONTINUE ERROR

Программа остановилась и не может быть продолжена командой CONT, которую можно использовать лишь после останова посредством CTRL-C или STOP при положении, что в программу не внесены изменения.

?DISPTYPE MISMATCH ERROR

Попытка выполнить команду вывода на экран в неправильном режиме. Например, DRAW или CHAR использованы в режиме TEXT или LORES. Другой пример - PLOT или PRINT использованы в режиме HIRES.

?DIVISION BY ZERO ERROR

Попытка делить на ноль. Проверьте, нет ли переменных со значением 0 или неопределенных переменных.

?FORMULA TOO COMPLEX ERROR

Это сообщение появляется в случае очень сложных формул. Разделите выражение на более мелкие части.

?ILLEGAL DIRECT ERROR

Попытка использовать в режиме прямого выполнения команду, которую можно задавать только в программе (например INPUT или GET).

?ILLEGAL QUANTITY ERROR

Использован параметр, значение которого выходит за допустимый интервал. Проверьте точно:

- какие действующие интервалы заданы при определении параметров;
- значения, являющиеся результатом вычисления выражений и участвующие как параметры в указанной строке;
- каждое использование INT, а также автоматическое округление (при превращении вещественных чисел в целые).

Это сообщение об ошибке следует и при присвоении значения, которое больше 32767 или меньше 32768.

?NEXT WITHOUT FOR ERROR

Программа встретила NEXT, для которого нет соответствующего FOR...TO. Причиной может быть пропуск FOR...TO, некорректное указание переменной цикла или неправильная передача управления в цикле.

?OUT OF DATA ERROR

Попытка прочитать командой READ несуществующую последовательность элементов, т.е. наличие слишком большого числа команд READ или указание в DATA недостаточного количества элементов.

?OUT OF MEMORY ERROR

Сообщение означает, что превышен размер оперативной памяти, предназначенной для хранения данных, программ на языке БЕЙСИК, переменных и т.д. В ряде случаев решением может стать освобождение неиспользуемых областей командой FRE.

Это же сообщение выдается и при использовании в программе больше 16 вложенных один в другой циклов или подпрограмм.

?OVERFLOW ERROR

Сообщение о переполнении. Получается, когда результат вычислений является слишком большим числом.

Самое большое число, которое может обрабатывать Правец-8Д, равно приблизительно 1.7E38, а самое маленькое - 2.93E-39.

?REDIM'D ARRAY ERROR

Сообщение выдается при попытке повторно определить данный массив в рамках одной программы независимо от того, как он был определен первоначально - командой DIM или по умолчанию.

?REDO FROM START

Использована команда INPUT, которая ожидает ввода с клавиатуры только числовых данных. Сообщение выдается при попытке ввести текст. Управление возвращается команде INPUT для повторной попытки ввода.

?RETURN WITHOUT GOSUB ERROR

Для оператора RETURN не существует соответствующего оператора GOSUB.

?STRING TOO LONG ERROR

Допустимая максимальная длина символьной строки равна 255 символам. Строка, введенная или полученная объединением строк, превысила этот предел.

?SYNTAX ERROR

Неправильный формат команды. Интерпретатор БЕЙСИК не распознает ее. При этом в общем случае имеется синтаксическая ошибка, например, неправильно написаны зарезервированные слова, ошибка в пунктуации и т.д.

?TYPE MISMATCH ERROR

Эта ошибка появляется при попытке присвоить числовой переменной или функции значение символьной строки и обратно. Сообщение указывает на попытку прямого обмена данными различного типа.

?UNDEF'D STATEMENT ERROR

Попытка передать управление строке с несуществующим номером.

?UNDEF'D FUNCTION ERROR

Сообщение о неопределенной функции.

Программа встретила выражение, включающее вычисление пользовательской функции, которая не определена предварительно командой DEF FN.

ТАБЛИЦА КОДОВ ASCII (КОИ-7)

Коды 0-31

Код	**Действие**	**Ввод с клавиатуры**
0	NULL	CTRL - @
1	копирование символа	CTRL - A
•2		CTRL - B
3	прерывание	CTRL - C
•4	символы двойной высоты	CTRL - D
•5		CTRL - E
6	озвученная клавиатура	CTRL - F
7	звуковой сигнал (BEL)	CTRL - G
•8	курсор влево (BS)	CTRL - H
•9	курсор вправо (TAB)	CTRL - I
•10	курсор вниз (LF)	CTRL - J
•11	курсор вверх (VT)	CTRL - K
12	стирание экрана (FF)	CTRL - L
•13	RETURN	CTRL - M
14	стирание строки	CTRL - N
15	блокировка экрана	CTRL - O

16	управление печатью (SYN)	CTRL - P
17	курсор включен	CTRL - Q
18		CTRL - R
19	экран	CTRL - S
20	верхний регистр (CAPS LOCK)	CTRL - T
21		CTRL - U
22		CTRL - V
23		CTRL - W
24	игнорирование строки (CAN)	CTRL - X
25		CTRL - Y
26		CTRL - Z
27	ESC (ESCAPE)	CTRL - [
28		CTRL - \
29		CTRL -]
30		CTRL - ^
31	CTRL-DEL	

Коды 32-127

В режиме LORES 0 изображаются стандартные символы, в режиме LORES 1 - альтернативные. В следующей таблице приведены символы стандартного изображения.

Код	Символ	Код	Символ	Код	Символ
32	пробел	64	@	96	Ю
33	!	65	A	97	А
34	"	66	B	98	Б
35	#	67	C	99	Ц
36	$	68	D	100	Д
37	%	69	E	101	Е
38	&	70	F	102	Ф
39	'	71	G	103	Г
40	(	72	H	104	Х
41	)	73	I	105	И
42	*	74	J	106	Й
43	+	75	K	107	К
44	,	76	L	108	Л
45	-	77	M	109	М
46	.	78	N	110	Н
47	/	79	O	111	О
48	0	80	P	112	П
49	1	81	Q	113	Я
50	2	82	R	114	Р
51	3	83	S	115	С
52	4	84	T	116	Т

53	5	85	U	117	У
54	6	86	V	118	Ж
55	7	87	W	119	В
56	8	88	X	120	Ь
57	9	89	Y	121	Ъ
58	:	90	Z	122	З
59	;	91	[	123	Ш
60	<	92	\	124	
61	=	93	]	125	Щ
62	>	94	^	126	Ч
63	?	95		127	DEL

УПРАВЛЯЮЩИЕ КОДЫ ESCAPE

Нажатие клавиши ESC подготавливает компьютер для ввода управляющего символа. В компьютере Правец-8Д таким образом определяется режим вывода символов в рамках одной строки. Для каждой строки экрана можно использовать свой режим изображения символов - различным цветом, на различном фоне, мигающие, двойной высоты и т.д. Второй символ, который вводится после ESC, определяет атрибут для вводимых вслед за ним символов. Комбинацию ESC и управляющего символа будем называть далее кодом ESCAPE.

Далее приведены управляющие коды ESCAPE. Они вводятся непосредственно с клавиатуры или из программы (с использованием PRINT CHR$(27); CHR$(n), где n является кодом второго символа). В середине таблицы даны значения атрибутов, которые можно использовать с командами PLOT (для режима LORES) и FILL (для режима HIRES). Если в них задается число вместо символа, интерпретатор БЕЙСИК рассматривает его как ASCII код. Таким образом кодами от 0 до 23 можно непосредственно задавать различные атрибуты в качестве последнего параметра. Избегайте использовать коды от 24 до 31, которые связаны с синхронизацией кадровой частоты монитора.

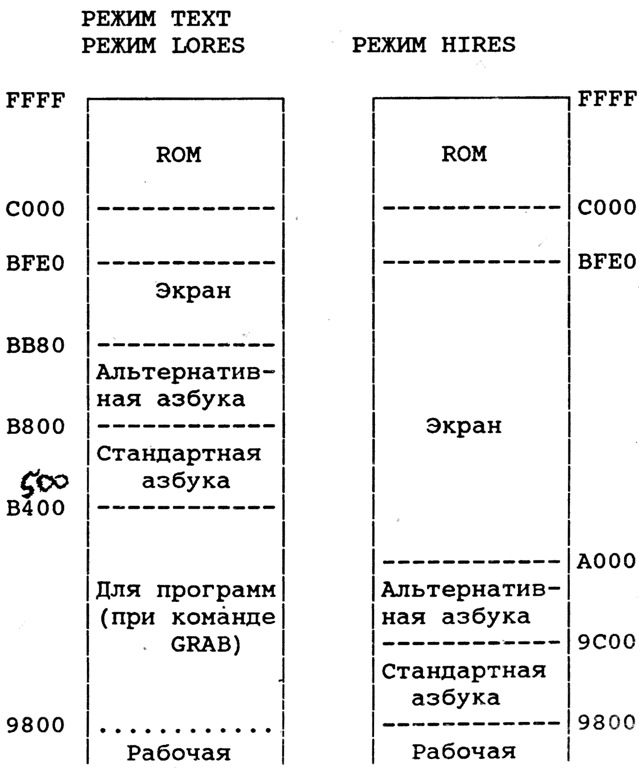
Атрибуты задаются в начале или в середине каждой строки. Ими определяется цвет "чернил" (сами символы) и "бумаги" (фон, на который выводятся символы).

Код	**Атрибут**	**Режим**
ESCAPE @	0	черные чернила
ESCAPE A	1	красные чернила
ESCAPE B	2	зеленые чернила
ESCAPE C	3	желтые чернила
ESCAPE D	4	синие чернила
ESCAPE E	5	лиловые чернила
ESCAPE F	6	светло-синие чернила
ESCAPE G	7	белые чернила
ESCAPE H	8	стандартная азбука
ESCAPE I	9	альтернативная азбука
ESCAPE J	10	стандартная азбука двойной высоты
ESCAPE K	11	альтернативная азбука двойной высоты
ESCAPE L	12	стандартная азбука - мигающий режим
ESCAPE M	13	альтернативная азбука - мигающий режим

ESCAPE N	14	стандартная азбука двойной высоты - мигающий режим
ESCAPE O	15	альтернативная азбука двойной высоты - мигающий режим
ESCAPE P	16	черная бумага (фон)
ESCAPE Q	17	красная бумага
ESCAPE R	18	зеленая бумага
ESCAPE S	19	желтая бумага
ESCAPE T	20	синяя бумага
ESCAPE U	21	лиловая бумага
ESCAPE V	22	светло-синяя бумага
ESCAPE W	23	белая бумага

ПАМЯТЬ

Распределение памяти компьютера Правец-8Д



```
      |   область    |        |   область    |
      |   программ   |        |   программ   |
      |              |        |              |
0500  |--------------|        |--------------| 0500
      |  Страница 4  |        |  Страница 4  |
0400  |--------------|        |--------------| 0400
      |  Страница 3  |        |  Страница 3  |
      | (адреса В/В) |        | (адреса В/В) |
0300  |--------------|        |--------------| 0300
      |  Страница 2  |        |  Страница 2  |
0200  |--------------|        |--------------| 0200
      |  Страница 1  |        |  Страница 1  |
      |    (стек)    |        |    (стек)    |
0100  |--------------|        |--------------| 0100
      |  Страница 0  |        |  Страница 0  |
0000  |______________|        |______________| 0000
```

Примечание: все числа в таблице являются шестнадцатеричными.

Память подразделяется на постоянную (ROM) и динамическую (RAM).

Постоянная память содержит интерпретатор БЕЙСИК и системные программы. Она занимает старшие адреса памяти от #C000 до #FFFF. Для компьютера Правец-8Д эта память включает 16384 байта, которые читаются практически непрерывно.

Программист не может изменять, но может читать содержимое постоянной памяти.

С адреса #0000 до #BFFF расположена RAM память, организованная следующим образом:

- 5 страниц по 256 байтов для системных нужд в младших адресах;
- память для пользователя (рабочая область программ);
- память для альтернативного и стандартного наборов символов;
- память для графического режима с высокой разрешающей способностью.

Рассмотрим более подробно организацию RAM памяти, с которой практически работает пользователь.

Страница 0 содержит текущую информацию о работе микропроцессора (например, адреса начала и конца выполняемой программы на языке БЕЙСИК, указатели к памяти с переменными и т.д.).

Страница 1 содержит адреса управления подпрограммами. Принято называть ее стек-памятью. В стек последовательно записываются, а затем извлекаются из него адреса возврата в циклы и подпрограммы.

Страница 2 предназначена для хранения некоторых системных переменных, необходимых для прослеживания операций интерпретатора БЕЙСИК (например, позиция курсора, текущий регистр для кириллицы или латиницы, включен или выключен звуковой сигнал и т.д.).

Страница 3 содержит адреса ввода-вывода между компьютером и внешними устройствами, а также адреса обмена данными между отдельными системными элементами компьютера.

Страница 4 резервирует адреса от #0400 до #04FF для системных нужд.

За пятью системными страницами следует область памяти, предназначенная для программ пользователя - адреса от #0500 до #97FF. Память заполняется, начиная с нижних адресов, снизу вверх, при этом сразу за занимаемой программой областью следует область, предназначенная для всех определенных переменных. Часть RAM памяти содержит описание изображения на экране. В отличие от стандартного текстового режима (TEXT), экран для режима с высокой разрешающей способностью (HIRES) требует дополнительного объема памяти. Эта область (от #9800 до #B3FF) может передаваться программе пользователя посредством команды GRAB. При этом, разумеется, нельзя использовать режим HIRES.

Обратная команда RELEASE восстанавливает область в качестве экранной памяти в режиме HIRES. В области памяти с адресами от #B400 до #BB7F содержится описание двух наборов символов компьютера Правец-8Д.

Стандартный набор содержит 128 символов и для него отведены 1024 байта. Альтернативный набор включает 112 символов и занимает 896 байтов. Каждый символ занимает 8 последовательных байтов, в которых определяется матрица точек, изображающая символ на экране. Заданные в альтернативном наборе символы могут быть изменены путем записи различных значений в адреса памяти, в которых описан соответствующий символ. Таким образом оформляются новые специальные символы, создаются различные шрифты.

После двух наборов символов расположена область, зарезервированная для описания экрана (от #BB80 до #BFDF). В конце остается еще одна небольшая область, предназначенная для программ на машинном языке и операций ввода-вывода (от #BFE0 до #BFFF).

БЕЙСИК предлагает несколько специализированных команд для работы с памятью. Например HIMEM определяет самый высокий адрес, доступный программе на языке БЕЙСИК. Команды POKE и DOKE, функции PEEK и DEEK служат для чтения и записи ячеек памяти.

Подпрограммы и адреса в постоянной памети

Записанные в постоянной памяти подпрограммы можно вызывать из программ на языке БЕЙСИК с помощью команды CALL. Для большинства из них достаточно загрузить соответствующие регистры микропроцессора, прежде чем выполнить переход к ним посредством инструкции JSR. Параметры подпрограмм, которые работают с графическими изображениями и звуком, записываются в область памяти с начальным адресом #2E0. Параметры представляются как 16-битовые числа со знаком. После своего выполнения подпрограмма помещает код ошибки в байт с адресом PARAMS+0. Перед вызовом подпрограммы посредством CALL в адрес PARAMS+0 необходимо записать 0. Все адреса далее являются шестнадцатеричными. Все подпрограммы используют аккумулятор A и индексные регистры X и Y (т.е. изменяют их содержание), если только явно не указано иное.

VDU Адрес: F77C (?)

Выводит символ на экран и перемещает курсор на одну позицию вправо.

Параметры: X=код выводимого символа
Результаты: нет.
Используемые регистры: нет

STOUT Адрес: F865

Выводит сообщение на строку состояния на экране (самая верхняя строка - адреса от BB80 до BBA7). Сообщение должно быть закончено символом NULL.

Параметры: A=адрес сообщения (младший байт)
У=адрес сообщения (старший байт)
X=горизонтальная позиция первого символа
Результаты: X=следующая позиция

GTORKB Адрес: EB78

Чтение символа с клавиатуры. Символ получается с текущей скоростью повторения, определенной байтами с адресами #24E и #24F. Байт #24E задает задержку обычно 30

миллисекунд, прежде чем соответствующий клавише символ начнет повторяться. При постоянно нажатой клавише частота повторения задается в #F (обычно тоже 30 миллисекунд). Чтобы получить символы с максимальной скоростью, запишите единицы в оба байта (24E и 24F).

Параметры: нет
Результаты: A=ASCII код символа
Флаг для знака +
Флаг для знака -
Используемые регистры: нет

PRTCHR Адрес:F5C1

Выводит символ на печатающее устройство.
Параметры: A=ASCII код символа
Результаты: нет

OUTLED Адрес: E74A

Выводит начальную синхронизирующую последовательность (9 символов с ASCII кодом #16,SYN) к кассетофону.
Параметры: нет
Результаты: нет
Примечание: Скорость обмена с кассетофоном задается байтом #24D. Скорость будет высокой (2400 бит/сек) при значении 0 и низкой (300 бит/сек) при значении, превышающем 0.

GETSYN Адрес:E735

Читает синхронизирующую последовательность байтов с кассетофона.
Параметры: нет
Результаты: нет
Используемые регистры: A,X

OUTBYT Адрес:E65E

Записывает байт на кассетофон.
Параметры: A=ASCII код символа
Результаты: нет
Используемые регистры: A

RDBYTE Адрес:E6C9

Читает байт с кассетофона.
Параметры: нет

Результаты: A=ASCII код прочитанного символа
Используемые регистры: A

CURSET Адрес:E0C8

Параметры: PARAMS+1 : значение X
PARAMS+3 : значение Y
PARAMS+5 : значение F
Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала.

CURMOV Адрес:FDFD

Параметры: PARAMS+1 : значение X
PARAMS+3 : значение Y
PARAMS+5 : значение F
Результаты: PARANS+0 - код ошибки 1 при значениях вне допустимого интервала

DRAW Адрес:F110

Параметры: PARAMS+1: значение X
PARAMS+3: значение Y
PARAMS+5: значение F
Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала
Используемые регистры: A,X

CHAR Адрес:F12D

Параметры: PARAMS+1: код ASCII символа
PARAMS+3: код азбуки - 0 для стандартной и 1 для альтернативной
PARAMS+5: значение F
Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала

CIRCLE Адрес:F37F

Параметры: PARAMS+1: радиус
PARAMS+3: значение F
Результаты: код ошибки 1 при значениях вне допустимого интервала

PATRN Адрес:F11D

Параметры: PARAMS+1: значение шаблона PATTERN

Результаты: код ошибки 1 при значениях вне допустимого интервала

Используемые регистры: A

POINT Адрес: F1C8

Параметры: PARAMS+1: значение X
PARAMS+3: значение Y

Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала
PARAMS+1: 0 - фон
1 - основной цвет

FILL Адрес: F268

Параметры: PARAMS+1: количество строк
PARAMS+3: число клеток
PARAMS+5: значения

Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала

PAPER Адрес: F204

Параметры: PARAMS+1: цвет

Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала

INK Адрес: F210

Парамеры: PARAMS+1: цвет

Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала

PING Адрес: FA97

Доступна по адресу.

Примечание: Все программы для звука используют те же самые процедуры для передачи параметров, что и графические параметры.

SHOOT Адрес: FAB5

Доступна по адресу.

EXPLD Адрес: FACB

Доступна по адресу.

ZAP Адрес: FAE1

Доступна по адресу.

KBBEEP Адрес: FB14

Доступна по адресу. Вызывает звуковой сигнал при нажатии на клавишу.

CONTBP Адрес: FB2A

Доступна по адресу. Вызывает звуковой сигнал при нажатии CTRL.

SOUND Адрес: FB40

Параметры: PARAMS+1: канал
PARAMS+3: период звука
PARAMS+5: сила звука
Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала

MUSIC Адрес: FC18

Параметры: PARAMS+1: канал (1-3)
PARAMS+3: октава (0-7)
PARAMS+5: нота (1-12)
PARAMS+7: сила звука (0-15)
Результаты: PARAMS+0: код ошибки 1 при значениях вне допустимого интервала

PLAY Адрес: FBD0

Параметры: PARAMS+1: канал тона
PARAMS+3: канал шума
PARAMS+5: модулирование
PARAMS+7: период модулирования
Результаты: PARAMS+0 - код ошибки 1 при значениях вне допустимого интервала

W8912 Адрес: F590

Вводит данные из регистра X в музыкальный синтезатор. Подпрограмма поддерживает клавиатуру доступной.
Параметры: A=обращение к синтезатору
X=исходные данные

Результаты: нет

Адреса страниц 0 и 2

Страница 0

LINWID	#0031	- ширина строки
TXTTAB	#009A-#009B	- начальный адрес программы
VARTAB	#009C-#009D	- начальный адрес для переменных
ARYTAB	#009E-#009F	- начальный адрес области для массивов
STREND	#00AD-#00A1	- конечный адрес области для символьных последовательностей
MEMSIZ	#00A6-#00A7	- конечный адрес динамической памяти (HIMEM)
CHRGET	#00E2-#00E7	- получает символ из программы
CHRGOT	#00E8	- инструкция LDA
TXTPRT	#00E9-#0DEA	- указатель к интерпретируемому символу
SKPSPC	#00EB-#00EE	- подпрограмма пропуска пробелов
QNUM	#00EF-#00F8	- проверка является ли символ цифрой
CHRRTS	#00F9	- инструкция RTS

Страница 2

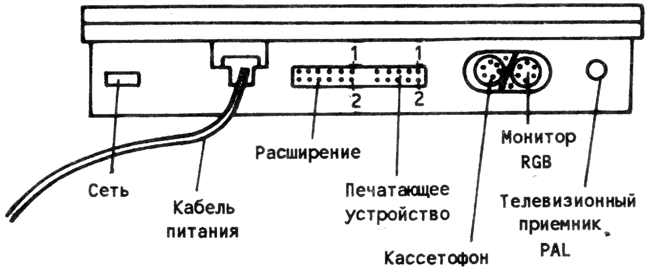
KEYAD	#0208	- адрес нажатой клавиши
KBSTAT	#0209	- выбор верхнего регистра #A4=левый SHIFT #A7=правый SHIFT
CAPLCK	#020C	- #80=только верхний регистр
PAT	#0213	- параметр PATTERN для команд CIRCLE и DRAW
CURX	#0219	- координата x для графического режима
CURY	#021A	- координата y
GRA	#021F	- режим экрана: 1=HIRES, 0=TEXT/LORES
XVDU	#238	- переход к подпрограмме VDU
XGETKY	#23B	- переход к подпрограмме ввода с клавиатуры (GTORKB)
XPRTCH	#23E	- переход к подпрограмме вывода на печатающее устройство (PRTCHR)
XSTOUT	#241	- переход к подпрограмме вывода

		строки состояния (STOUT)
INTFS	#244	- переход к подпрограмме обработки прерываний
NMIJP	#247	- переход к подпрограмме обработки немаскируемых прерываний
INTSL	#24A	- возврат из подпрограммы обработки прерываний
TSPEED	#24D	- скорость чтения/записи на кассетофон: 0 - 2400 бит/сек >0 - 300 бит/сек
KBDLY	#24E	- время задержки перед автоматическим повторением нажатой клавиши
PWIDTH	#256	- ширина строки печатающего устройства (стандартно 80)
VWIDTH	#257	- ширина строки экрана (стандартно 40)
CURROW	#268	- текущая строка курсора
MODEO	#26A	- биты этого байта определяют текущее состояние различных функций: 7 - не используется 6 - не используется 5 - 1 = защищены столбцы 0 и 1 на экране 4 - 1 = последний выведенный символ ESC 3 - 1 = выключает звуковой сигнал 2 - не используется 1 - 1 = включено VDU 0 - 1 = включен курсор
BGND	#026B	- код цвета фона (бумага)+16
FGND	#026C	- код цвета символов (чернила)
CURON	#0270	- флаг включения/выключения курсора
CURINV	#0271	- флаг смены курсора
TIMER1	#0272-#0273	- таймер клавиатуры
TIMER2	#0274-#0275	- таймер курсора
TIMER3	#0276-#0277	- 16-битовый таймер с шагом 0,01 секунды; используется командой WAIT
VDUL2	#0278-#0279	- адрес второй строки экрана
VDUL1	#027A-#027B	- адрес первой строки экрана

VDUCH	#027C-#027D	-	число символов, которые смещаются при прокрутке: обычно 26 строк x 40 символов
NOROWS	#027E	-	число строк на экране
ICM.AR	#02DF	-	код последней нажатой клавиши
PARAMS	#02E0	-	буфер для параметров подпрограммы звука и графического изображения

Примечание. Таймер отчитывает время в сторону уменьшения (в обратную сторону по сравнению с хронометром).

СХЕМЫ РАЗЪЕМОВ



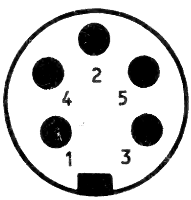
РАЗЪЕМ ДЛЯ РАСШИРЕНИЯ

+5V	A10	A9	A8	A7	A6	A5	D5	A2	A1	A0	A3	D2	RW	10	02	MAP
33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
GND	A11	A12	A13	A14	A15	D7	A4	D4	D3	D6	D1	Dd	IRQ	I/O	RST	ROMDIS

РАЗЪЕМ ДЛЯ ПЕЧАТАЮЩЕГО УСТРОЙСТВА

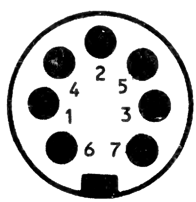
ACK	D7	D6	D5	D4	D3	D2	D1	D0	STB
19	17	15	13	11	9	7	5	3	1
20	18	16	14	12	10	8	7	5	2
				GND					

РАЗЪЕМ ДЛЯ МОНИТОРА



1. Красный
2. Зеленый
3. Синий
4. Синхронизация
5. Земля

РАЗЪЕМ ДЛЯ КАССЕТОФОНА



1. Выход
2. Земля
3. Вход
4 и 5. Звук
6 и 7. Реле

Литература

1. Сираков, П.А., О.П. Вълчев. Ключ за компютър. С., Народна младеж, 1985.
2. Петров, П.Д. БЕЙСИК за персонални компютри. С.,НУЦ, 1985.

АЗБУЧНЫЙ УКАЗАТЕЛЬ

С О Д Е Р Ж А Н И Е

Домашний компьютер Правец-8Д

Авторы: Орлин Петров Вылчев, инж. Пенчо Ангелов
Сираков, Димитр Христов Вавов

Национальность болгарская

Издание первое

Код 3205-18-88
Изд. № 16537
Редактор болгарского текста инж. Нина Денева
Редактор перевода инж. Румяна Ковачева
Художник Любомир Михайлов
Корректор Росица Стоянова
Технический редактор Иван Георгиев
Набор сделан в СК "Правец-Программа"
Подписано к печати 1.XII.1988 г.
Выход из печати 15.XII.1988 г.
Формат 60x84/16
Печ.л. 7
Уч.-изд. л. 6,53
Тираж 50000 + 112 экз.

Государственное издательство "Техника" - София,
Болгария
Государственная типография имени Атанаса Стратиева -
Хасково